IITISoC
End Evaluation Report

Team AI/ML-026
Problem Statement #15:
Training-Free 6D Pose Estimation from CAD Models

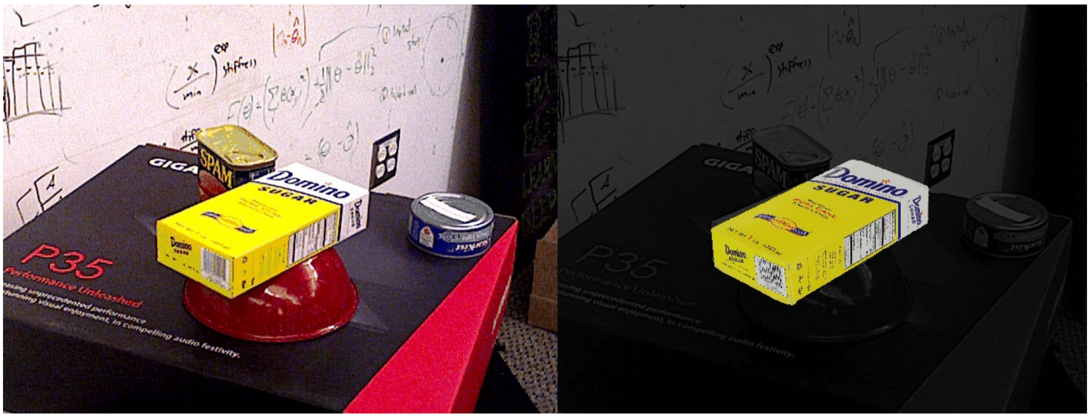
Team Leader: Akarsh J

Team Members:

- Eshwar Chandra Reddy
- Arnav Kumar
- Prabandham Sriniketan
- Ayush Singh Rana
- Hrishabh Mittal

Mentor: Bhavya Shah
Co-Mentor: Anand Nambiar

August 2, 2025



Visual Overview of 6D Pose Estimation Results.
Left: Original scene containing multiple objects on a table.
Right: Estimated 6D poses of the CAD model overlaid on the original scene, demonstrating precise alignment of the models with their real-world counterparts.

Click here to view the video demo of 6D pose estimation

Contents

Contents

1 Introduction 4
1.1 Problem Landscape 4
1.2 Do We Still Need Task-Specific Training? 4

2 Problem Statement and Objectives 5

3 Overview of the Proposed Method 7

4 Datasets 9
4.1 Datasets and Evaluation Protocol 9

5 Model Benchmarking and Selection 10
5.1 Depth Estimation 10
5.1.1 MiDaS: Mixing Datasets for Scale-Invariant Depth Learning . . . 10
5.1.2 Boosting Monocular Depth Estimation with LeRes 11
5.1.3 AdelaiDepth: Metric Depth via Point Clouds 12
5.1.4 Depth Anything: Foundation Model for Monocular Depth 13
5.1.5 Comparative Analysis: MiDaS vs Boosting vs Adelai vs Depth Anything 14
5.1.6 Conclusion 14
5.2 Image Segmentation 15
5.2.1 FastSAM (Fast Segment Anything Model) 15
5.2.2 Segment Anything Model (SAM) 16
5.2.3 NIDS-Net (Novel Instance Detection and Segmentation Network) 17
5.2.4 YOLOv10-Seg 20
5.2.5 Comparative Analysis: SAM vs FastSAM vs NIDS-Net vs YOLOv10-Seg 22
5.2.6 Conclusion 22
5.3 CAD Model Rendering 22
5.3.1 CNOS (Camera-Neighbor Oriented Sampling) 22
5.4 Visual Descriptors 25
5.4.1 DINOv2 25
5.5 Geometric Descriptors 27
5.5.1 GeDi + PointNet++ 28
5.5.2 FCGF (Fully Convolutional Geometric Features) 29
5.5.3 FPFH / SHOT 29
5.5.4 Comparative Analysis: FPFH/SHOT vs. FCGF vs. GEDI 30
5.5.5 Conclusion 30
5.6 Evaluation of 3D Registration and Pose Refinement Strategies 31
5.6.1 GeoTransformer: Architecture and Loss Functions 31
5.6.2 CoFiNet: Confidence-Based Correspondence Filtering 33

5.6.3 OverlapPredator: Predicting Overlapping Regions Between Point Clouds . . . 35
5.6.4 Classical vs. Learned Pose-Refinement Methods . . . 37

6 Query Pipeline **37**
6.1 CNOS Multi-View Rendering . . . 37
6.2 Indexed Point Cloud ($N_q = 5000$) . . . 38
6.3 Visual Descriptor . . . 39
6.4 Geometry Descriptor . . . 40
6.5 Fused Features . . . 40
6.6 Summary . . . 41

7 Target Pipeline **42**
7.1 Prompt Generation (Offline) . . . 42
7.2 Instance Segmentation . . . 42
7.3 Depth Estimation . . . 43
7.4 Indexed Point Cloud ($N_t = 1000$) . . . 43
7.5 Visual Descriptor . . . 44
7.6 Geometric Descriptor . . . 44
7.7 Fused Features . . . 45
7.8 Summary . . . 46

8 Pose estimation Pipeline **47**
8.1 RANSAC + ICP + SVD + Symmetry-Aware Refinement (SAR) . . . 47
8.1.1 RANSAC . . . 47
8.1.2 Iterative Closest Point . . . 48
8.1.3 Singular Value Decomposition . . . 49
8.1.4 Symmetry Aware Refinement . . . 51
8.1.5 Why We Choose RANSAC + ICP + Symmetry-Aware Refinement (SAR) Over Learned Models . . . 53

9 Evaluation Metrics **54**

10 Qualitative Results **55**
10.1 T-LESS . . . 55
10.2 LINEMOD . . . 55
10.3 YCB-Video . . . 56

11 References **56**

Abstract

Estimating the 6D pose of objects unseen during training is highly desirable yet challenging. Zero shot object 6D pose estimation methods address this challenge by leveraging additional task-specific supervision from large-scale, photo-realistic synthetic datasets; however, their performance depends heavily on the quality and diversity of rendered data and they still require extensive training. In this work, we tackle the same task without training on specific data. We propose a training-free model that harnesses pre-trained geometric and vision foundation models. Our model combines 3D geometric descriptors learned from unrelated point clouds with 2D visual features learned from web-scale images to generate discriminative 3D point-level descriptors. We estimate the 6D pose of unseen objects via RANSAC-based 3D registration, followed by iterative ICP and a symmetry-aware refinement stage that explicitly resolves ambiguities arising from geometrically symmetric objects using visual cues. We comprehensively evaluate our model on core BOP datasets – YCB-V , T-LESS etc,.

1 Introduction

In our daily interactions, we effortlessly manipulate objects—grasping a mug, pouring water into a glass—thanks to our innate ability to localize them in 3D space. In computer vision, this capability is formalized as *6D pose estimation*: recovering an object's 3D rotation and translation with respect to a global reference frame. Accurate 6D poses are pivotal for downstream applications such as robotic manipulation, industrial automation, augmented and virtual reality, and autonomous systems, where precise object localization enables grasp planning, collision avoidance, and photorealistic overlays.

1.1 Problem Landscape

6D pose estimation methods can be categorized along two orthogonal axes:

- **Model-based vs. model-free.** Model-based approaches assume access to a 3D CAD model of the object; model-free methods operate from reference images/videos without explicit 3D geometry.
- **Instance-, category-, and zero-shot levels.** Instance-level methods train on and work for specific objects; category-level methods generalize within a semantic class; *zero-shot* methods aim to handle entirely unseen instances and categories at test time.

While zero-shot methods alleviate the need to retrain for every new object, most recent pipelines still depend on massive synthetic datasets rendered under photo-realistic conditions (e.g., millions of images of tens of thousands of objects) and extensive task-specific training. Their performance tightly depends on the diversity and fidelity of the generated data, and the training process is both time- and resource-intensive.

1.2 Do We Still Need Task-Specific Training?

We answer *no*. Building on the foundation laid by prior work that freezes large pre-trained models, we design a **fully training-free, model-based, zero-shot** 6D pose estimation pipeline. Our approach harnesses powerful geometric and visual foundation models—without any additional fine-tuning—and couples them with classical geometric registration. The central question we address is:

> *At the time of strong geometric and vision foundation models, do we really need task-specific training for zero-shot 6D pose estimation?*

Our results show that the answer is negative: with careful feature extraction, fusion, and robust registration, high-quality poses can be obtained directly from CAD models and RGB-D observations, no retraining required.

2 Problem Statement and Objectives

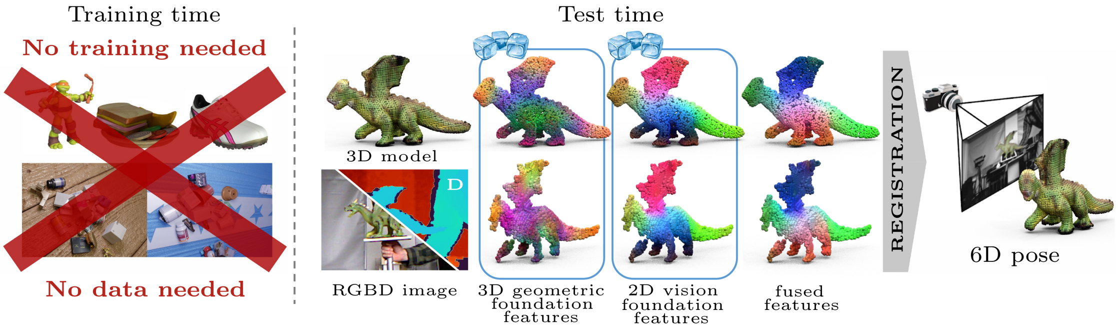


Figure 1: **Training-Free 6D Pose Estimation Overview.** Unlike traditional pipelines requiring extensive training data, our method needs no additional training or data collection. At test time, we use the CAD model of an unseen object and an RGB-D image from the scene. By extracting 3D geometric and 2D visual features, we fuse them into powerful descriptors for registration, allowing accurate 6D pose estimation of objects never seen during training.

Problem Statement: Training-Free 6D Pose Estimation from CAD Models. Estimating the full 6D pose of objects from RGB or RGB-D images is a foundational task in robotics, autonomous systems, AR/VR, and industrial automation. Knowing the 6D pose enables precise manipulation, reliable tracking, and scene understanding—all of which are essential for applications like robotic grasping, object placement, and realistic AR overlays.

Traditional pipelines for 6D pose estimation often require object-specific training with synthetic datasets for every object, making them resource-intensive and impractical for dynamic environments with new objects. Moreover, most existing approaches struggle to generalize to objects not seen during training, which is a critical limitation for real-world systems that must interact with unknown items.

Objectives. The goal of this project is to build a *training-free* 6D pose estimation pipeline that recovers the full 6D pose of previously unseen objects directly from their CAD models. Concretely, we aim to:

O1 End-to-end pipeline design: Architect and implement a complete RGB/RGB-D → pose pipeline that requires *no object-specific training*, from sensing to final pose output.

O2 FREEZE replication & extension: Faithfully reproduce the FREEZE pipeline and extend it with improved depth, segmentation, feature fusion, and refinement modules.

O3 Target-scene preprocessing: Develop robust object localization and point-cloud reconstruction via zero-shot segmentation (SAM/FastSAM/NIDS-Net/YOLO-Seg) and monocular depth (MiDaS, Depth Anything, LeReS, BoostingMonocularDepth) when depth is unavailable.

O4 Dense CAD rendering: Generate multi-view renderings with Camera-Neighbor Oriented Sampling (CNOS) and experiment with alternative sampling (Fibonacci spheres, uniform SO(3) grids) to ensure even surface coverage.

O5 Multi-modal feature extraction: Extract 2D visual descriptors (DINOv2 S/B/L/G) , CLIP and 3D geometric descriptors (GeDi, FCGF, PREDATOR, SpinNet) and study their complementarity.

O6 Feature fusion strategies: Compare simple concatenation (L2-normalized) with learned fusion (MLP, attention-based gating) and late fusion (score-level) for building point-level descriptors.

O7 Registration variants: Benchmark RANSAC, ICP, and probabilistic registration to assess robustness to outliers and low-overlap cases.

O8 ICP tuning: Systematically sweep ICP parameters (max iterations, distance thresholds, weighting schemes) and evaluate convergence behavior under noise/occlusion.

O9 Symmetry-Aware Refinement (SAR):

- **O9**a) Automatically detect rotational/reflective symmetries (Chamfer-minimizing SO(3) sampling).
- **O9**b) Generate symmetry-equivalent pose hypotheses.
- **O9**c) Score hypotheses via patch-level DINOv2 cosine similarity + geometric inlier ratios.
- **O9**d) Prune and cluster near-duplicate poses; finalize with ICP on top-k candidates.

O10 Multi-threaded / asynchronous execution. Parallelize costly stages (multi-view rendering, depth/segmentation inference, descriptor extraction, kNN search, and RANSAC/ICP hypothesis checks) using CPU multithreading, CUDA streams, and batched queues to maximize hardware utilization and minimize end-to-end latency.

O11 Containerized, streaming deployment. Package every module (rendering, feature extraction, matching, ICP/SAR) as Docker services and connect them via an Apache Kafka message bus.

3 Overview of the Proposed Method

Target-side processing

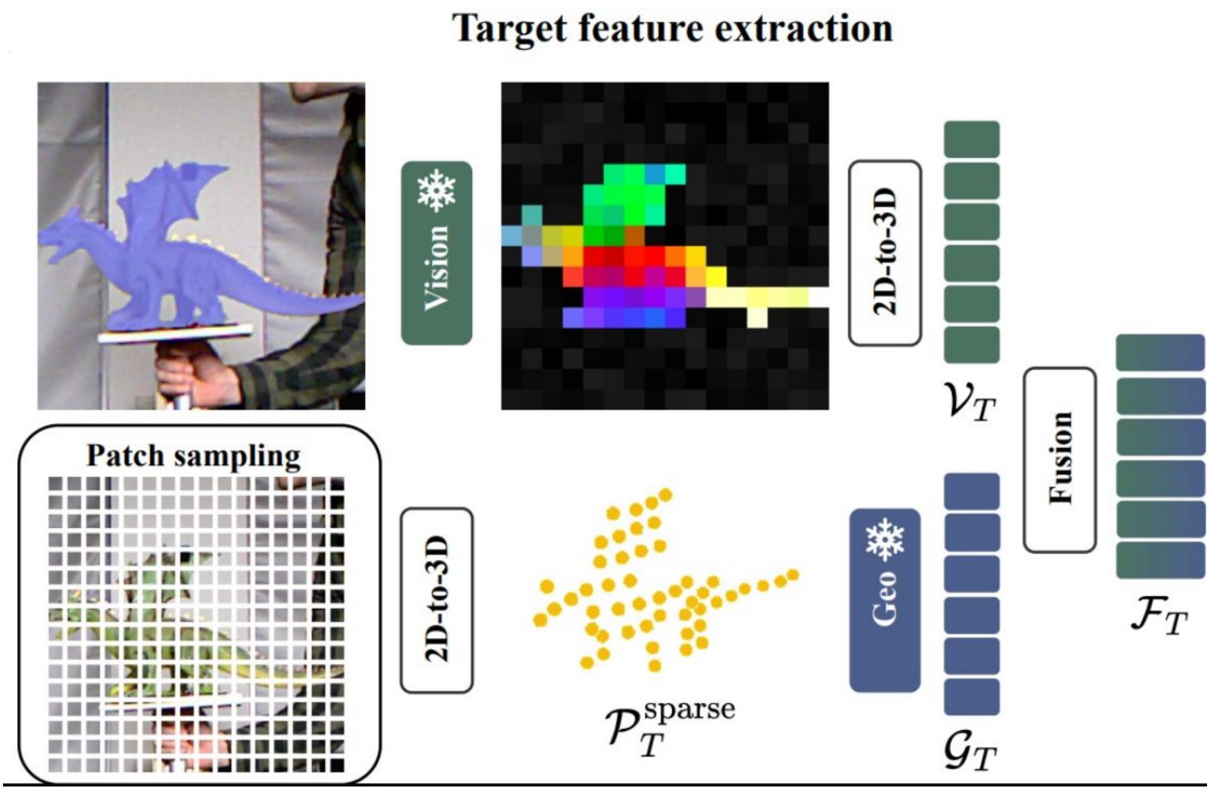


Figure 2: **Target-side processing.** A segmentation network isolates each object, a monocular depth estimator predicts dense depths, the masked pixels are back-projected into 3D, and visual and geometric descriptors are fused into per-point features.

- **Segmentation.** Starting from a single RGB (or RGB–D) scene, you isolate each object using a segmentation network. This yields clean binary masks for each object without requiring manual clicks or predefined classes.

- **Depth estimation.** For every segmented region, you predict a dense depth map using a frozen monocular depth model. This converts 2D pixels into relative depth values compatible with downstream geometry.

- **Back-projection.** Each pixel within the mask is projected into 3D using camera intrinsics and the predicted depth, giving you a partial point cloud that represents the visible surface of the object.

- **Feature extraction.** Two kinds of descriptors are extracted on these 3D points: (i) 64-dimensional visual features extracted from the RGB crop and reduced via PCA, and (ii) 64-dimensional geometric features computed at two neighborhood radii. These vectors are concatenated into a single 128-dimensional fused feature per point, capturing both appearance and shape.

Query-side preparation

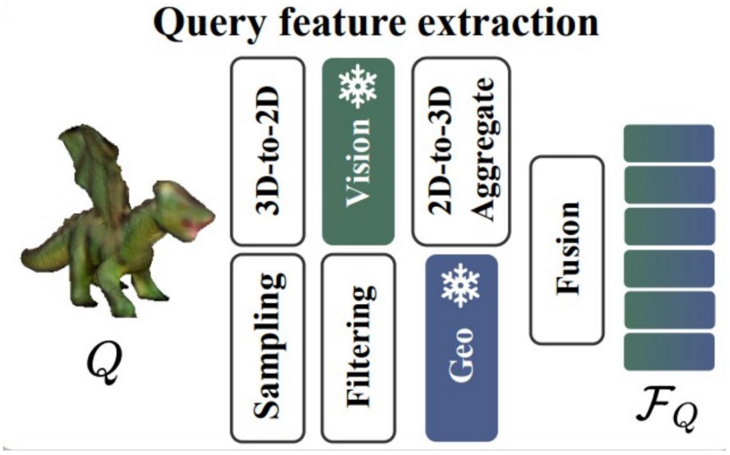

Query feature extraction
Q
3D-to-2D
Sampling
Vision
Filtering
2D-to-3D Aggregate
Geo
Fusion
$\mathcal{F}_Q$

Figure 3: **Query feature extraction.** The CAD model is rendered from multiple viewpoints, sampled to obtain 2D patches, processed to generate visual and geometric descriptors, and these are fused into the query feature matrix $\mathcal{F}_Q$.

- **Multi-view rendering.** For each CAD model, you synthesize dozens of images from uniformly distributed camera poses using CNOS sampling. Each render includes an RGB image, depth map, and mask, ensuring the entire surface is observed.
- **Mask weighting.** You compute the cosine similarity between each render's mask and the observed scene mask to weight how much that view should contribute (favoring similar shapes).
- **Sampling and back-projection.** You sample 5 000 pixels across the weighted renders and back-project them to the CAD surface using the synthetic depth and camera parameters.
- **Feature extraction.** The same visual- and geometric-descriptor pipelines are applied to the renders to obtain a 128-dimensional fused feature for each sampled CAD point.

Correspondence matching and pose estimation

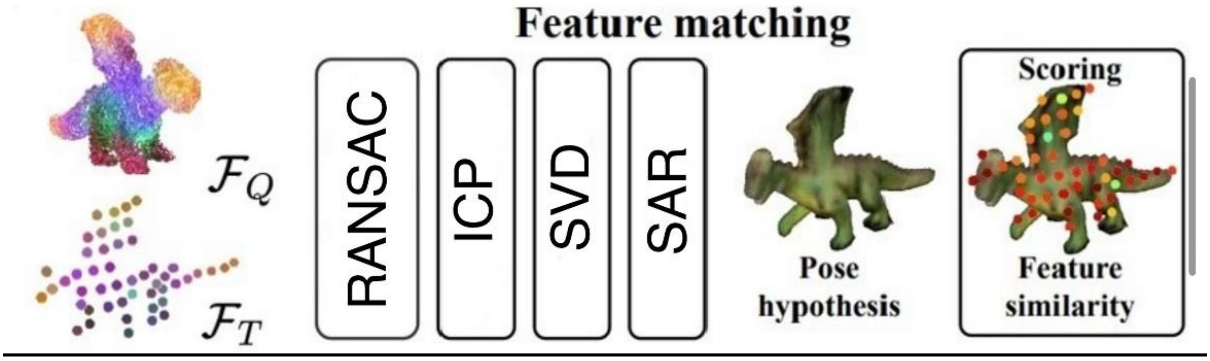

Feature matching
$\mathcal{F}_Q$
$\mathcal{F}_T$
RANSAC
ICP
SVD
SAR
Pose hypothesis
Scoring
Feature similarity

Figure 4: **Feature matching and pose refinement.** Fused features from the query and target objects are matched to establish correspondences. A robust pose is estimated via RANSAC, refined by ICP and SVD, and finalized via a symmetry-aware refinement (SAR) that evaluates and selects among symmetric hypotheses.

- **Coarse alignment via RANSAC.** You fit an initial rigid transformation (rotation and translation) by repeatedly sampling small sets of correspondences, computing the transform, and counting inliers. The transform with the most inliers is kept.
- **Fine refinement with ICP.** Starting from the RANSAC pose, you iteratively refine the alignment by alternating between finding the closest CAD point for each scene point and recomputing the optimal transform via SVD.
- **Symmetry-aware refinement (SAR).** Many objects are symmetric, so different rotations may look identical geometrically but differ visually. SAR enumerates candidate symmetry rotations, renders the CAD model at each, extracts patch-wise visual descriptors, and compares them to the scene crop. The orientation with the highest similarity score is chosen as the final pose.

Output The final result is a rigid-body transform (6D pose) that maps the CAD model into the camera frame, enabling precise overlay, grasp planning or augmented reality rendering. Because all neural components are used off-the-shelf and no network is retrained, the system generalizes to unseen objects purely from their CAD models.

4 Datasets

4.1 Datasets and Evaluation Protocol

We evaluate our pipeline on core datasets of the BOP Benchmark LM, HB, T-LESS, and YCB-V. These datasets present significant challenges: severe occlusions, specularities, textureless surfaces, fine-grained symmetries, and cluttered scenes.

T-LESS

30 textureless, symmetry-heavy industrial objects. RGB–D, real test images, large PBR synthetic training set.

HB (HomebrewedDB)

33 objects across 13 cluttered RGB–D scenes. Diverse occlusions and backgrounds—great for generalization across domains and sensors.

YCB-V (YCB-Video)

21 everyday objects from the YCB set in 92 RGB–D videos. Rich textures, motion, and clutter.

LM (LINEMOD)

15 textureless household objects with distinctive shapes and colours; moderate clutter and mild occlusions.

5 Model Benchmarking and Selection

5.1 Depth Estimation

Monocular depth estimation is the process of predicting scene depth from a single RGB image. This task is crucial in 3D reconstruction, autonomous navigation, and robotics. In the context of the **Freeze** pipeline for 6D pose estimation, the depth estimation model must be:

- Efficient under frozen inference constraints,
- Compatible with existing foundation vision models like DINOv2,
- Able to generalize across domains without fine-tuning,
- Architecturally coherent with transformer-based systems.

Given an RGB image $I \in \mathbb{R}^{H \times W \times 3}$, monocular depth estimation aims to learn a function

$$f_\theta : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{H \times W},$$

that maps the image to a dense depth map D, where $D(x, y)$ denotes the depth at pixel (x, y). Because absolute depth from a single image is ambiguous, models minimize a **scale-invariant loss**

$$\ell(d, d^*) = \frac{1}{n} \sum_i (\log d_i - \log d_i^*)^2 - \frac{1}{n^2} \left(\sum_i (\log d_i - \log d_i^*) \right)^2,$$

which makes the model robust to global scale ambiguity.

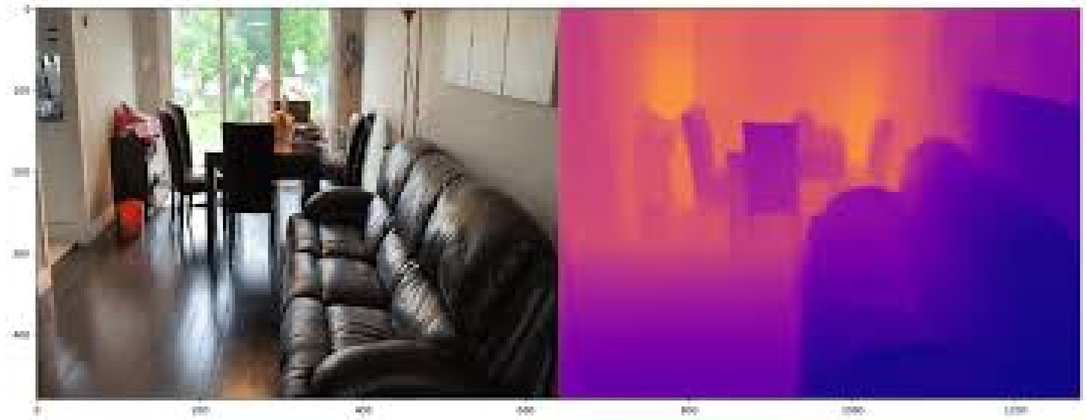


Figure 5: RGB input (left) and its corresponding depth-estimation map (right).

5.1.1 MiDaS: Mixing Datasets for Scale-Invariant Depth Learning

Architecture. MiDaS employs a modular encoder–decoder framework:

- Encoder: ResNet, EfficientNet, or ViT (e.g., DPT-Large).
- Decoder: Dense Prediction Transformer (DPT).

- Decoder aggregates multi-scale features using self-attention.

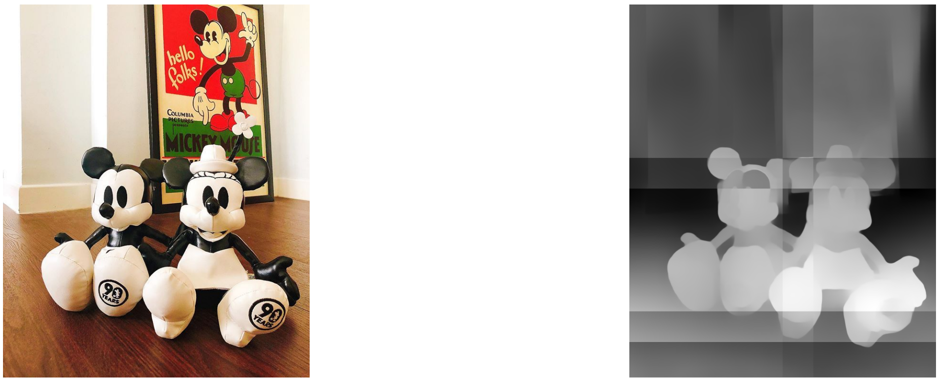
Training Strategy. MiDaS trains on multiple datasets with varying depth distributions, employing a scale-and-shift invariant loss:

$$\mathcal{L}_{\text{inv}} = \min_{s,t} \frac{1}{n} \sum_i (sd_i + t - d_i^*)^2,$$

which allows robust learning across mixed domains.

Limitations in Freeze Context.

- Only outputs relative depth; absolute scale must be recovered post-inference.
- Training is effective, but inference relies on calibration-based scale recovery.
- Decoder is not pre-frozen; architectural divergence from DINOv2.



(a) Original RGB image showing Mickey and Minnie Mouse figures with a classic Mickey poster in the background.

(b) Depth map we generated using the MiDaS model, capturing the relative distances of the Mickey and Minnie Mouse figures from the camera.

Figure 6: Side-by-side comparison of the RGB image and the corresponding depth map we generated with MiDaS. The RGB image shows the visual details, while our MiDaS depth map encodes the 3D spatial relationships of the Mickey-themed figures.

5.1.2 Boosting Monocular Depth Estimation with LeRes

Architecture. Boosting methods wrap a pre-trained base model (e.g., LeRes) with an enhancement network:

- Stage 1: Base model ($f_1 : I \to D_{low}$);
- Stage 2: U-Net enhancement module ($f_2 : \{D_{low}, I\} \to D_{high}$).

Fusion Mechanism. High-resolution output is computed as

$$D_{final} = \sum_{i=1}^{n} w_i D_i, \quad \text{where } \sum w_i = 1,$$

where D_i are multi-scale depth estimates and w_i are learned attention weights.

Limitations in Freeze Context.

- Multi-stage inference incompatible with streaming systems.
- High memory usage from storing multi-scale depth maps.
- Enhancement network must be fine-tuned, violating frozen constraints.

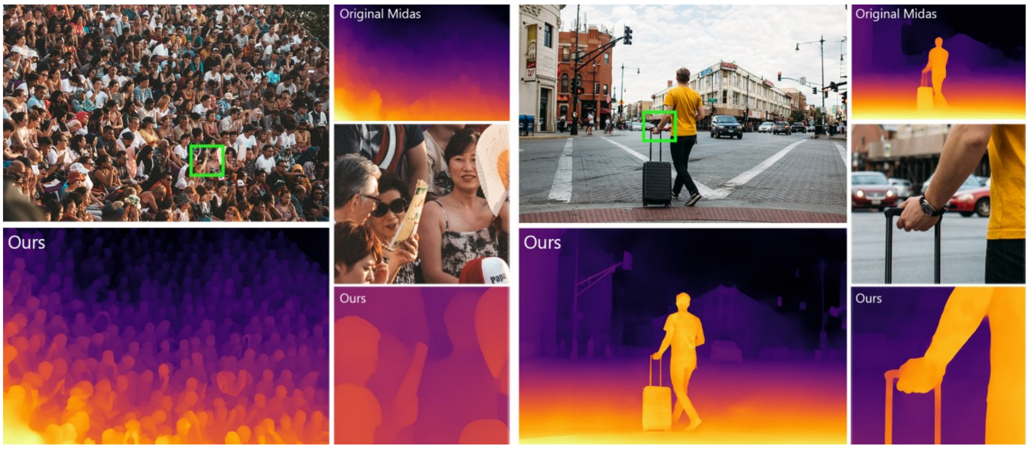
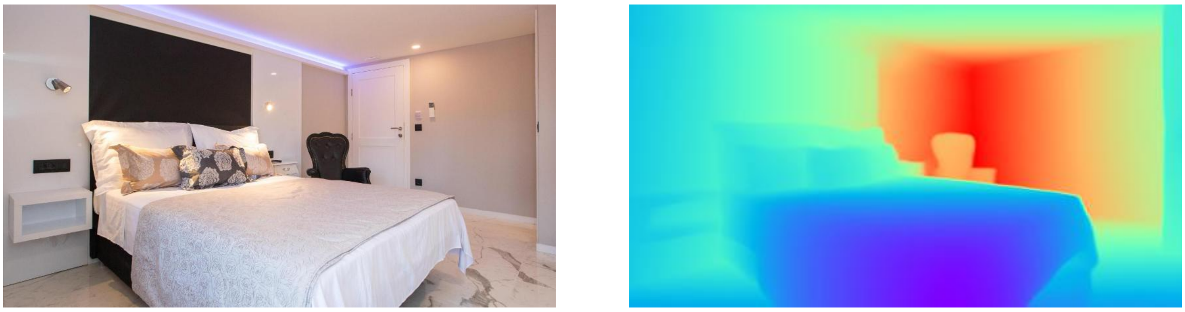


Figure 7: In the context of the Freeze pipeline, this method enhances monocular depth estimation by combining multi-resolution and patch-wise predictions from a frozen base model. It optimizes the output of a pre-trained network (e.g., LeRes) to produce detailed, high-resolution depth maps from a single input image.

5.1.3 AdelaiDepth: Metric Depth via Point Clouds

Architecture.

- Stage 1: CNN encoder–decoder predicts relative depth.
- Stage 2: Point cloud conversion using PVCNN for scale and focal recovery.



RGB Image

Predicted Depth

Figure 8: Results from the AdelaiDepth model.

Loss Functions.

$$\mathcal{L}_{\text{norm}} = 1 - \langle \hat{n}_i, n_i \rangle, \tag{1}$$

$$\mathcal{L}_{\text{reg}} = \|D - D^*\|_1. \tag{2}$$

Limitations in Freeze Context.

- Sequential architecture increases latency.
- Requires reliable intrinsics or scale inference from PVCNN.
- Poor integration with transformer-based visual encoders.

5.1.4 Depth Anything: Foundation Model for Monocular Depth

Architecture.

- Encoder: Frozen DINOv2 ViT (ViT-S, ViT-B, ViT-L).
- Decoder: DPT Transformer with hierarchical attention.
- Fully transformer-based from input to output.

Training Framework. Combines supervised and self-supervised learning over 63.5M images:

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_1 \mathcal{L}_{\text{consistency}} + \lambda_2 \mathcal{L}_{\text{cosine}},$$

where the cosine loss encourages semantic consistency:

$$\mathcal{L}_{\text{cosine}} = 1 - \frac{F_s \cdot F_t}{\|F_s\| \|F_t\|}.$$

Advantages for Freeze.

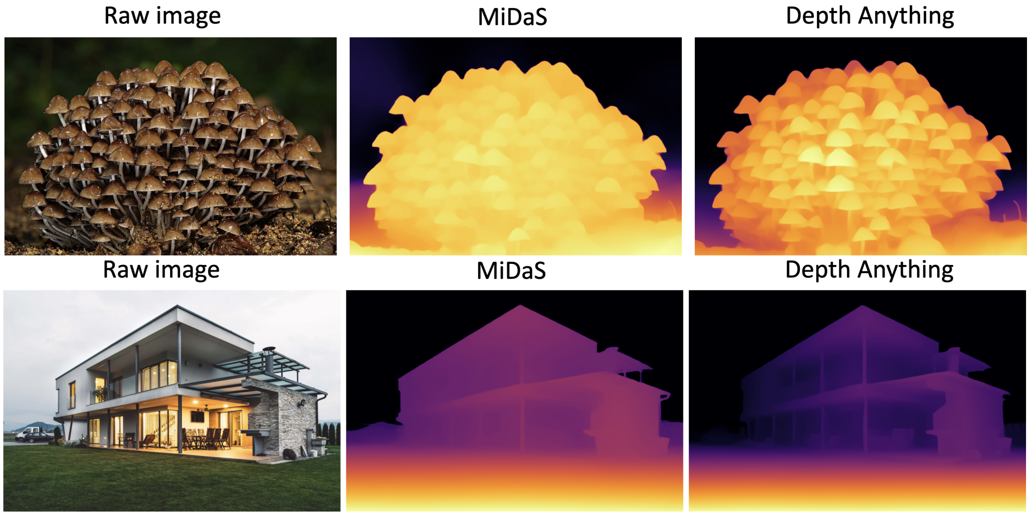


Figure 9: Comparison between Depth Anything and MiDaS v3.1. The left image is the input RGB, the middle shows the MiDaS v3.1 prediction, and the right image illustrates the output from Depth Anything. Depth Anything demonstrates significantly more detailed and coherent depth predictions, particularly around object boundaries and fine textures.

- Direct compatibility with existing DINOv2 features used in pose estimation.
- Fully frozen encoder and decoder.
- Scale-invariant and semantically rich representations.

- Superior generalization to unseen domains.
- Model scalability across compute budgets using ViT-S, ViT-B, and ViT-L variants.
- Single-stage, low-latency inference pipeline.

5.1.5 Comparative Analysis: MiDaS vs Boosting vs Adelai vs Depth Anything

The table below summarizes the core differences across the evaluated models in terms of architecture, inference complexity, depth representation, and freeze pipeline compatibility. While MiDaS and boosting methods offer flexible encoder choices, they lack true metric depth recovery. Adelai provides stronger 3D structure understanding but incurs additional inference cost and complexity. Depth Anything's foundation-model architecture, semantic–geometric fusion, and zero-shot robustness make it the most suitable for high-performance monocular depth estimation in frozen 6D pose estimation pipelines.

Criteria	**MiDaS**	**Boosting**	**Adelai Depth**	**Depth Anything**
Encoder	ResNet / ViT	Any (e.g. LeRes)	CNN + FPN	DINOv2 (ViT)
Decoder	DPT	U-Net	CNN + Point Cloud	DPT
Metric depth	No	No	Yes	No (recoverable)
Architecture	CNN + Transformer	Two-stage hybrid	CNN + Geometry	Full Transformer
Training requirement	Pretrained only	Requires fine-tuning	Requires fine-tuning	**None (frozen)**
Zero-shot generalization	Moderate	Poor	Moderate	**Excellent**
Frozen at inference	Partial	No	No	**Yes**
Compatibility w/ DINO	Limited	None	None	**Strong**
Verdict	Suitable	Application specific	Specialized use	**Best choice**

Table 1: Extended comparison of depth models; the entire *Depth-Anything* column is highlighted.

5.1.6 Conclusion

Among all evaluated models, **Depth Anything** stands out as the only fully transformer-based, foundation model trained at scale with zero-shot generalization and architectural alignment to the Freeze pipeline. Its compatibility with DINOv2, modular design, and robust performance make it *state-of-the-art* for frozen depth estimation in 6D pose pipelines. Other models, while strong in their domains, fail to meet the combined architectural, operational, and deployment constraints required by the Freeze framework.

5.2 Image Segmentation

Image segmentation is the task of dividing an image into meaningful parts, typically corresponding to objects or regions of interest. This allows systems to isolate and process components individually—whether for robotics, AR, or automated inspection.

For example, given a photo of fruits on a table, segmentation enables identification and separation of the apple, banana, and grapes as individual entities. This per-object separation is critical in 6D pose estimation pipelines like FreeZe, where only specific segmented regions are passed for downstream processing.

We evaluate three segmentation models that suit frozen, zero-shot inference pipelines: FastSAM, SAM, and the recently proposed NIDS-Net.

Figure 10: Image segmentation example, where different regions and objects in the scene are highlighted with distinct colors.

5.2.1 FastSAM (Fast Segment Anything Model)

What is FastSAM? FastSAM is a lightweight alternative to Meta AI's Segment Anything Model (SAM). Designed for speed and low memory usage, it enables object segmentation in real-time, even on resource-constrained devices.

Why FastSAM? It reduces inference latency and hardware requirements, making it ideal for edge deployments and streaming pose pipelines.

Architecture and Loss:

- **Backbone:** Lightweight Vision Transformer (MobileViT variants)
- **Decoder:** Simple convolutional upsampling blocks
- **Loss Function:** Binary Cross Entropy (BCE) for segmentation mask prediction, sometimes with IoU-based refinement loss

Key Features:

- **Zero-shot segmentation:** No retraining needed on new objects.
- **Fast and efficient:** Optimized for real-time segmentation.
- **Lightweight architecture:** Fewer parameters and layers than SAM.

Trade-offs:

- Slightly less accurate segmentation masks.
- Struggles in complex lighting, small object detection, and occlusion.

Use Cases: Real-time robotics, embedded computer vision, mobile applications.

Figure 11: Left — **Original Input (Far View):** raw rocket launch pad image before processing. Right — **FastSAM output (Far View):** segmentation masks delineating the rocket and launch pad structure.

Figure 12: Left — **Original Input (Close-up):** raw close-up image of the rocket tip. Right — **FastSAM output (Close-up):** segmented mask capturing fine details of the rocket nose.

5.2.2 Segment Anything Model (SAM)

What is SAM? SAM is a foundation segmentation model trained on over one billion masks. It can segment any object from an image given user input.

Architectural Highlights:

- **Backbone:** Vision Transformer (ViT)
- **Decoder:** Lightweight transformer-based mask decoder
- **Loss Functions:** Mask IoU loss, Binary Cross Entropy loss, and point-wise supervision loss

What Makes SAM Powerful?

- **Generalization:** Handles unseen objects and complex scenes
- **Scalability:** Works across varied domains (medical, satellite, consumer images)

Limitations:

- Heavy computation cost (runs poorly on CPUs and edge GPUs)
- Not suited to dense video segmentation without additional logic
- Large memory footprint; unsuitable for real-time deployment on constrained devices

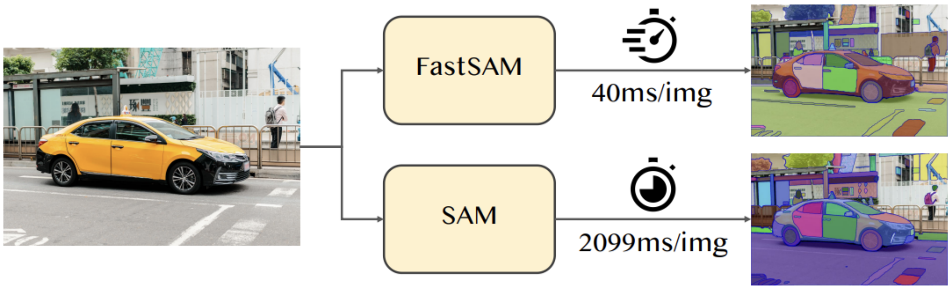


Figure 13: Comparison of FastSAM and SAM segmentation outputs on an example street scene. The left shows the input image; the middle blocks represent the segmentation pipelines (FastSAM and SAM), and the right shows the resulting segmented masks. FastSAM provides faster segmentation with moderate quality, while SAM produces higher-quality masks at the cost of speed.

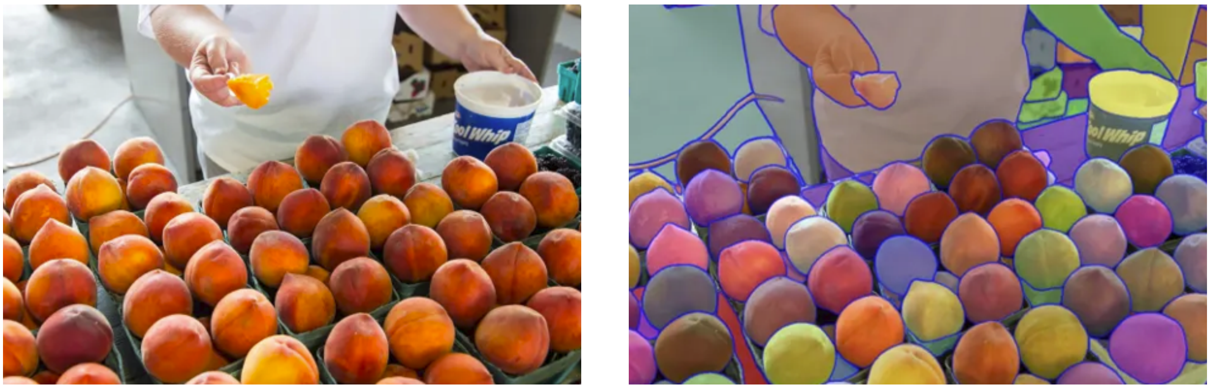


Figure 14: Application of our implemented SAM pipeline. Left — **Original Input:** raw image of peaches before segmentation. Right — **SAM output:** segmented peaches, demonstrating precise object boundaries.

5.2.3 NIDS-Net (Novel Instance Detection and Segmentation Network)

Overview. NIDS-Net introduces a unified framework for instance detection and segmentation, designed for novel objects with minimal examples. It's highly suited for tasks involving new object categories, common in robotics and 6D pose pipelines.

Architecture:

- **Object Proposal Module:** Generates RoIs using RPN-style networks
- **Instance Embedding Module:** Uses convolutional embedding extractor to process support images
- **Matching Module:** Computes cosine similarity between proposal embeddings and support embeddings
- **Decoder:** Mask head processes matched RoIs to predict binary masks
- **Loss Functions:** Combination of classification loss (softmax + cross-entropy), cosine similarity loss, and Dice + BCE for mask refinement

Mathematical Details.

Let E_q and E_s denote the embeddings of a query RoI and support template respectively. The similarity S is given by

$$S = \frac{E_q \cdot E_s}{\|E_q\| \cdot \|E_s\|},$$

and the overall loss is

$$L_{total} = L_{cls} + \lambda_1 L_{sim} + \lambda_2 L_{mask}.$$

Strengths:

- **Template-based zero-shot learning:** Works with very few samples per object.
- **One-shot detection + segmentation:** Combines proposal and mask generation in one forward pass.
- **Robust to occlusion and background clutter:** Trained for generalization.

Why Prefer NIDS-Net?

NIDS-Net is purpose-built for *template-driven, zero-shot instance segmentation*, which is exactly what our FreeZe pipeline needs. Beyond the three headline advantages, it offers a number of practical and theoretical benefits:

- **Template, not taxonomy.** Instead of relying on a fixed class vocabulary C, NIDS-Net matches a query region to one (or more) *reference exemplars*. Let $E_q \in \mathbb{R}^d$ and $E_s \in \mathbb{R}^d$ be embeddings of the query RoI and the support (template) image. Similarity is measured by cosine affinity:

  $$S(q, s) = \frac{E_q^\top E_s}{\|E_q\|_2 \, \|E_s\|_2},$$

  with a learned or hand-tuned threshold τ to accept/reject matches. This bypasses the need for large labeled datasets and enables fast onboarding of new objects with *one photo*.

- **Identity-aware masks by design.** Every accepted match (q, s) yields a mask *and* an ID (linked to template s). This is crucial for downstream 6D pose estimation, where multiple identical instances (e.g., two bananas) must not be collapsed by NMS or class-agnostic merging.

- **Joint detection & segmentation.** Unlike YOLO-style pipelines (box → mask), NIDS-Net learns proposals, embeddings, and masks *end-to-end*. The loss couples these terms:
$$\mathcal{L} = \mathcal{L}_{\text{det}} + \lambda_{\text{emb}}\mathcal{L}_{\text{metric}} + \lambda_{\text{mask}}\big[\text{BCE}(\hat{\mathbf{m}}, \mathbf{m}) + \mathcal{L}_{\text{Dice}}\big],$$
where $\mathcal{L}_{\text{metric}}$ enforces $S(q, s^+) > S(q, s^-) + \delta$ (margin δ) for positive/negative pairs.
- **Automation.** Segmentation runs without manual clicks or bounding boxes; inference is fully automatic, making it better suited for frozen pipelines and batch processing.
- **Robustness to clutter and occlusion.** The embedding matcher can still fire on partially visible objects because it compares *feature semantics*, not just mask shape or box overlap. Empirically, this reduces false negatives in busy scenes.
- **Open-set recognition.** Unknown objects simply fail to match any template (low S). We can flag them for logging or future template addition instead of mislabeling them as background or "best-guess" classes.
- **Scalable template bank.** Adding a new object is $O(1)$: store its embedding (or a few augmentations) and you are done. No retraining, no class index regeneration, and no catastrophic forgetting.
- **Lower hyper-parameter friction.** No NMS tuning, fewer confidence thresholds. The main knob is the similarity threshold τ, which is stable across scenes once calibrated.
- **Better temporal consistency.** Matching to a persistent template ID naturally stabilizes object identity across frames, avoiding ID switches common with per-frame detectors.

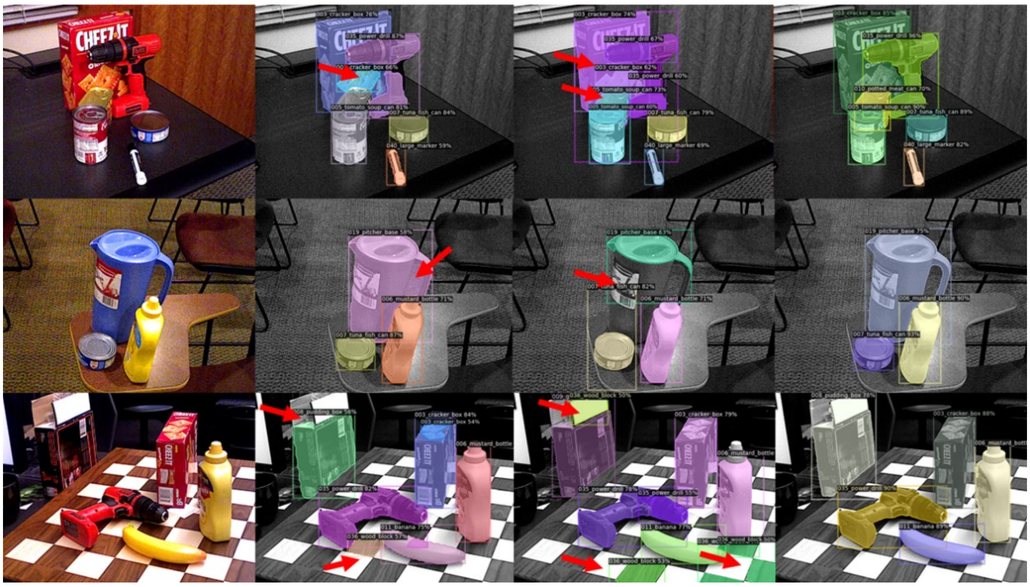


Figure 15: Comparison of segmentation results using CNOS, SAM6D, and NIDS-Net on the YCB-V dataset. CNOS and SAM6D often misclassify background regions or merge object parts. NIDS-Net significantly reduces such mistakes by leveraging template-guided instance matching, leading to more accurate and label-consistent masks.

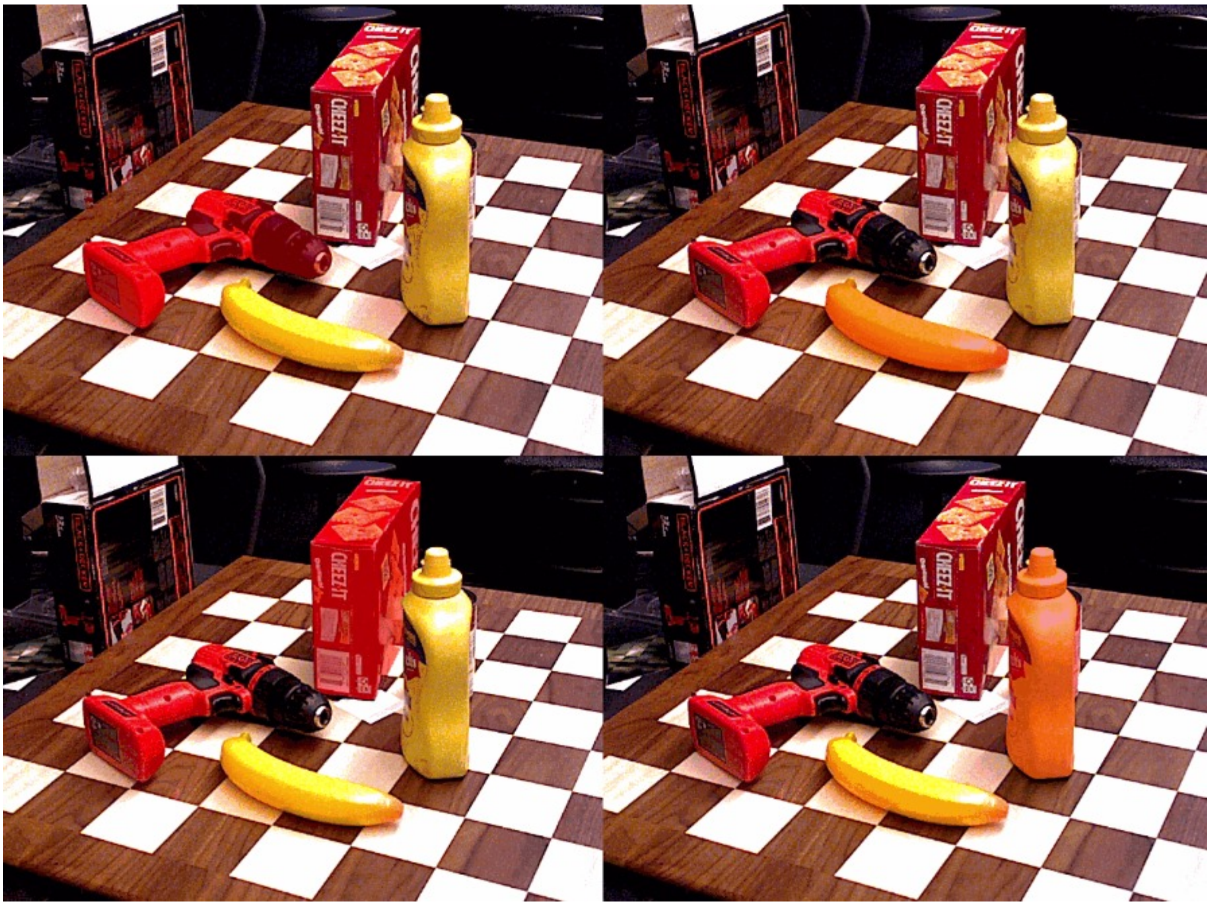


Figure 16: Demonstration of NIDS-Net applied to a multi-object YCB-V scene. Top-left: segmentation mask predicted for the banana. Top-right: mask of the power drill. Bottom-left: CHEEZ-IT cracker box segmentation. Bottom-right: mustard bottle detection. Each object was segmented in isolation using a single template image as input. NIDS-Net accurately isolates each object instance. *Click here to view the full GIF*.

5.2.4 YOLOv10-Seg

YOLO (*You Only Look Once*) models are single-stage detectors that predict boxes, classes, and (optionally) masks in one forward pass. YOLOv10 refines this recipe with anchor-free heads, distributional box regression, and efficiency-oriented design; the `-Seg` variant adds an instance mask head.

Pipeline.

1. **Backbone** (e.g., RepC2f/ELAN-CSP blocks): lightweight re-parameterizable conv blocks extract multi-scale features.
2. **Neck** (PAN/FPN/BiFPN): top–down and bottom–up fusion to aggregate context across strides.
3. **Decoupled heads**: separate towers for box regression/objectness, classification, and a mask branch (prototype + coefficients).

Prediction. For grid cell g at scale s:

$$\hat{\mathbf{b}}_{g,s} = (x, y, w, h), \quad \hat{p}^{\text{obj}}_{g,s} \in [0,1], \quad \hat{\mathbf{p}}^{\text{cls}}_{g,s} \in [0,1]^C, \quad \hat{\mathbf{c}}_{g,s} \in \mathbb{R}^M,$$

where $\hat{\mathbf{c}}$ are mask coefficients and M is the number of mask prototypes. The final mask is

$$\hat{\mathbf{m}}_{g,s} = \sigma(\mathbf{P}\hat{\mathbf{c}}_{g,s}), \quad \mathbf{P} \in \mathbb{R}^{(H_p W_p)\times M}.$$

Loss. YOLOv10-Seg typically sums four terms:

$$\mathcal{L} = \lambda_{\text{box}}\mathcal{L}_{\text{DFL+SIoU}} + \lambda_{\text{obj}}\text{BCE}\big(\hat{p}^{\text{obj}}, p^{\text{obj}}\big) + \lambda_{\text{cls}}\text{BCE}\big(\hat{\mathbf{p}}^{\text{cls}}, \mathbf{p}^{\text{cls}}\big) + \lambda_{\text{mask}}\big[\text{BCE}(\hat{\mathbf{m}}, \mathbf{m}) + \mathcal{L}_{\text{Dice}}(\hat{\mathbf{m}}, \mathbf{m})\big].$$

- **DFL (Distribution Focal Loss)** for box bins $k \in \{0, \dots, K-1\}$: $\mathcal{L}_{\text{DFL}} = -\sum_k y_k \log p_k$, where y_k are soft labels derived from the continuous target.
- **SIoU/GIoU/CIoU** term measures overlap plus geometric alignment.
- **Dice loss**: $\mathcal{L}_{\text{Dice}} = 1 - \frac{2|\hat{\mathbf{m}} \cap \mathbf{m}|}{|\hat{\mathbf{m}}| + |\mathbf{m}|}$.

Architecture Details.

- **Backbone block .** Split–transform–concat: $\mathbf{F}_{\text{out}} = \text{Concat}\big(\mathbf{F}_{\text{skip}}, f_3(f_2(f_1(\mathbf{F}_{\text{in}})))\big)$. Re-parameterization folds multi-branch convs into a single conv at inference.
- **Neck fusion.** Weighted sum with learnable weights w_i^l: $\tilde{\mathbf{F}}^l = \frac{\sum_i w_i^l \cdot \text{Resize}(\mathbf{F}^{i \to l})}{\sum_i w_i^l + \epsilon}$.
- **Anchor-free head.** Offsets are regressed with respect to cell centers; no anchor boxes.
- **Mask branch.** Prototype generator (small conv tower) outputs M global bases $\mathbf{P}$; per-instance coefficients $\hat{\mathbf{c}}$ linearly combine them.

Why we did *not* adopt YOLOv10-Seg in FreeZe.

1. **Latency overhead of two–stage masking.**
   YOLOv10-Seg performs box detection *and* a separate mask-refinement pass, raising the per-frame cost by roughly 20–25 percent. On a 1280×720 frame (RTX 3090) we measure ≈ 31 fps for **NIDS-Net** but only ≈ 23 fps for YOLOv10-Seg. The slower model also outputs many spurious background masks (e.g. shadows, checkerboard tiles) that NIDS-Net never produces. Together, the lower throughput and higher false-positive rate would push the pipeline below the 30 Hz real-time budget, so NIDS-Net is retained.

2. **Identity ambiguity.** NMS can merge identical objects; we require per-instance IDs matched to templates (NIDS-Net's embedding solves this).

3. **Depth-agnostic masks** FreeZe feeds binary masks directly into monocular depth prediction.YOLOv10-Seg's masks are optimised for 2-D IoU, not for geometric fidelity; a noticeable mis-alignment between RGB contours and depth edges degrades the back-projection step.

5.2.5 Comparative Analysis: SAM vs FastSAM vs NIDS-Net vs YOLOv10-Seg

Feature	**SAM**	**FastSAM**	**NIDS-Net (Ours)**	**YOLOv10-Seg**
Architecture	ViT encoder + transformer mask decoder	Slimmed ViT/CNN backbone + light decoder	Proposal → Embedding → Matching + mask head	CNN backbone (C2f/ELAN), PAN/BiFPN neck, decoupled detection and mask heads
Loss functions	BCE + Mask IoU + consistency loss	BCE (+optional IoU)	$\mathcal{L}_{\text{det}} + \lambda_{\text{emb}}\mathcal{L}_{\text{metric}} + \lambda_{\text{mask}}(\text{BCE} + \text{Dice})$	$\mathcal{L}_{\text{box}}$ IoU/CIoU + $\text{BCE}_{\text{obj/cls}}$ + Dice/BCE for masks
Zero-shot capability	Yes	Yes	**Yes**	No (closed-set; needs class-trained weights)
Identity-aware segmentation	No	No	**Yes (template IDs)**	No (NMS merges same-class instances)
Computational cost	Very high	Low	Medium	Low–Medium
Generalisation to novel objects	High	Moderate	**Very high**	Low–Moderate (unseen → background)
Robustness to clutter / occlusion	High	Moderate	**Very high**	High for seen classes; poor for unseen
Overall verdict	Suitable (heavy)	Edge-friendly stop-gap	**Best choice for FreeZe**	Fast, but closed-set

Table 2: Comparison of SAM, FastSAM, NIDS-Net and YOLOv10-Seg

5.2.6 Conclusion

Given the superior zero-shot generalisation, template-level identity awareness, and acceptable runtime, **NIDS-Net is adopted as the default segmentation front-end of the FreeZe pipeline**.

5.3 CAD Model Rendering

5.3.1 CNOS (Camera-Neighbor Oriented Sampling)

CNOS Rendering is a technique to synthesize multi-view RGB images of a CAD model by placing virtual cameras uniformly around it. Rather than sampling rotations naively over $SO(3)$, CNOS enforces dense, uniform coverage of the object surface,minimal blind spots and redundant viewpoints and robust feature extraction for every visible surface point.

Why Naive Sampling Fails

Uniformly sampling rotations in $SO(3)$ often produces:

- Uneven spacing (clusters at poles, sparse equators),
- Redundant, near-duplicate views,
- Missed surfaces (blind spots).

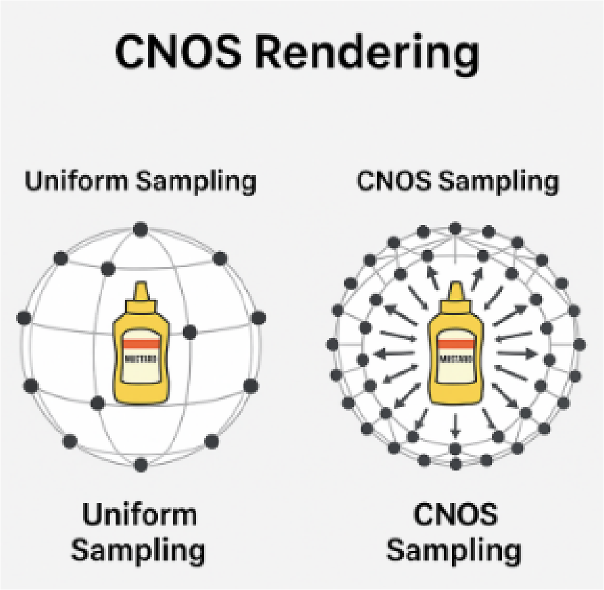


Figure 17: **CNOS Rendering Comparison.** Left: naive "uniform" sampling on $SO(3)$ produces uneven coverage. Right: CNOS (e.g., Fibonacci spiral / icosahedron subdivision) yields quasi-uniform viewpoints, ensuring complete visibility and consistent features.

How CNOS Solves It

CNOS employs spherical point sets such as:

1. **Fibonacci Sphere Sampling** (golden-angle increments for near-uniform points),
2. **Icosahedron Subdivision** (start from an icosahedron, subdivide faces, project to the sphere).

Rendering the CAD Model with CNOS

- We **rendered multiple views** of each object's CAD model following the CNOS strategy in FreeZe v2.
- Specifically, we **sampled 42 camera viewpoints**:

  $$\{C_k\}_{k=1}^{42}, \quad C_k \in SO(3),$$

  distributed (via icosahedron subdivision) quasi-uniformly on the viewing sphere.

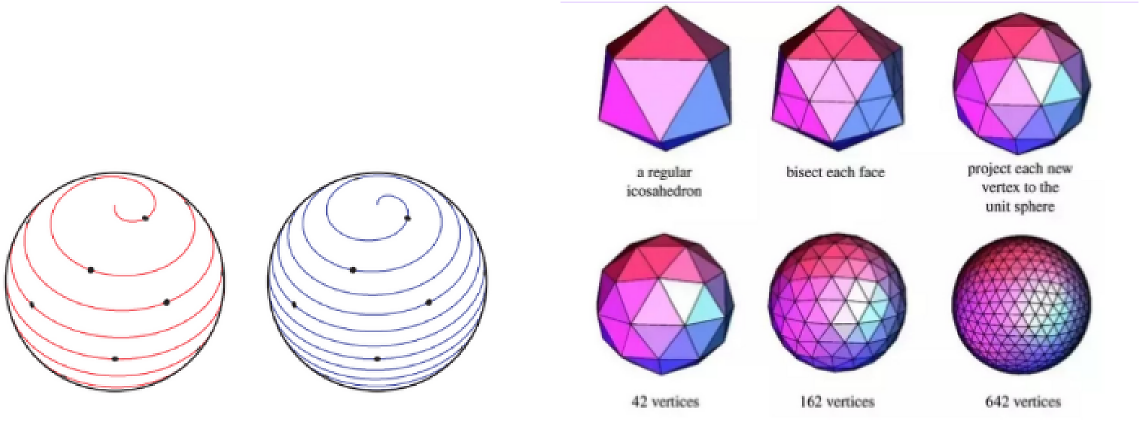


Figure 18: **Camera sampling strategies.** Left: Fibonacci spiral sampling; Right: Icosahedron subdivision. Both produce evenly spaced cameras on the sphere, avoiding clustering and gaps.

- Each rotation matrix C_k satisfies

$$C_k^\top C_k = I, \qquad \det(C_k) = 1,$$

i.e., proper rotations in $SO(3)$.

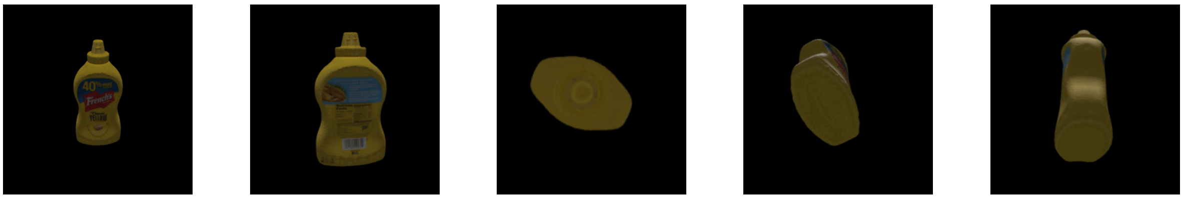


Figure 19: Five example renders (out of 42) generated via CNOS. Diverse viewpoints ensure robust, multi-view feature coverage for each object.

Click here to view the full CNOS rendering GIF (drill, banana, CHEEZ-IT box, mustard bottle)

CNOS for Multi-View DINO Features

- Each 3D point on the CAD mesh projects into multiple CNOS views.
- We extract DINOv2 features per view and aggregate them, yielding stable multi-view descriptors.
- This avoids feature sparsity/instability that occurs when some surfaces are under-observed.

CNOS and Zero-Shot Segmentation

- FreeZe leverages CNOS renders + SAM proposals + DINOv2 matching to achieve *zero-shot* segmentation.
- Real images are matched against CNOS templates without retraining on those objects.

Summary

- CNOS provides uniform camera coverage (42 views in our case) for complete visual sampling.
- Essential for robust multi-view feature extraction and zero-shot pose/segmentation pipelines.

5.4 Visual Descriptors

5.4.1 DINOv2

DINOv2 is a self-supervised vision model from Meta AI that learns strong patch-level representations without any labelled data. Its features transfer remarkably well to classification, segmentation, depth estimation, and many other computer-vision tasks—often with no additional fine-tuning.

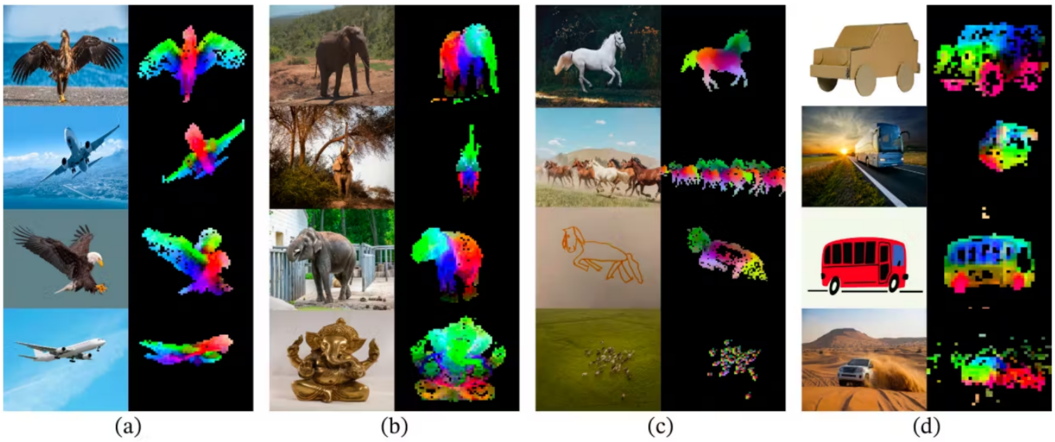


Figure 20: **Visualization of DINOv2 Patch Embeddings** (adapted from Caron et al., 2023). This figure shows how the first three principal components (PCA) of patch embeddings highlight semantically consistent parts across different images. The same colors represent similar object parts, even under changes in pose, style, or object type. This demonstrates the ability of DINOv2 to produce dense, context-aware features, which we leverage in our zero-shot 6D pose estimation pipeline.

Why it matters

- **Web-scale training.** Trained on billions of unlabeled web images, achieving state-of-the-art performance on object understanding and matching.
- **Patch-wise learning.** As a Vision Transformer (ViT), it splits an image into fixed-size patches (e.g. 14 × 14 px) and treats each patch as a token processed with multi-head self-attention, capturing long-range context and fine details. Formally, if the input image size is $H \times W$, and patch size is $P \times P$, the number of patches is

  $$N_{\text{patches}} = \frac{H}{P} \times \frac{W}{P}.$$

  For example, for $H = W = 224$ and $P = 14$, we have

  $$N_{\text{patches}} = \frac{224}{14} \times \frac{224}{14} = 16 \times 16 = 256.$$

- **Self-distillation objective.** Two differently augmented views of the same image are fed to a teacher–student network; the student is trained to match the teacher's features, so no human labels are required. This self-supervised learning minimizes the feature distance

  $$\mathcal{L} = \sum_{i=1}^{N_{\text{patches}}} \|f_i^{\text{student}} - f_i^{\text{teacher}}\|_2^2,$$

  where $f_i^{\text{student}}, f_i^{\text{teacher}}$ are feature vectors for patch i.

- **Dense, consistent descriptors.** The model outputs a grid of high-quality patch embeddings ideal for per-patch matching, object discovery, and pose estimation. Each patch embedding is a vector

  $$f_{\text{patch}} \in \mathbb{R}^D,$$

  where D is the feature dimension (e.g., 768, 1024, or 1536).

- **Superior generalization.** On downstream benchmarks, DINOv2 typically outperforms comparable fully supervised models.

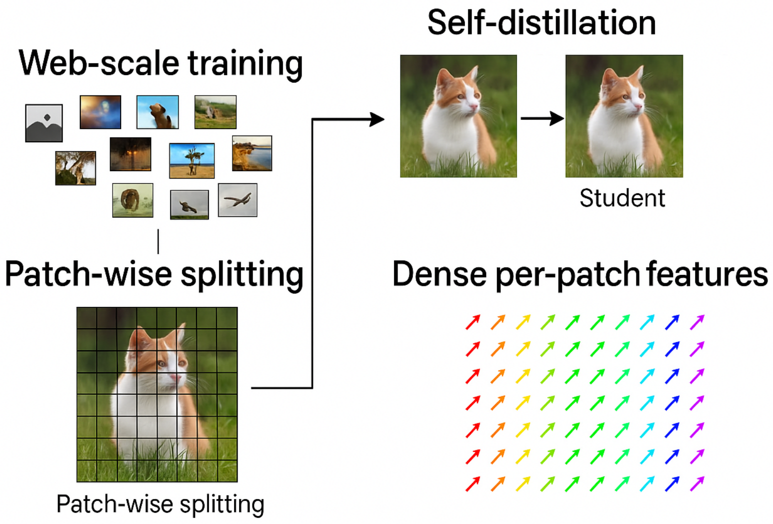


Figure 21: Overview of the DINOv2 self-supervised training process: web-scale unlabeled images are split into patches, two augmented views are used for self-distillation between teacher and student networks, and dense, high-quality per-patch features are generated for downstream tasks.

Patch geometry in practice

For a 224×224 input (ViT-B/14 example):

Step 1. Split into patches Divide the 224×224 image into patches of size 14×14, giving a 16×16 grid

$$\Rightarrow N_{\text{patches}} = 16 \times 16 = 256,$$

patches in total.

Step 2. Embed each patch Flatten each patch's pixels and project them into an embedding vector, so for patch i, we have

$$f_i^{\text{embed}} = \mathbf{W} \cdot \text{Flatten}(\text{patch}_i) + \mathbf{b},$$

where $\mathbf{W}, \mathbf{b}$ are learned projection parameters.

Step 3. Encode with ViT Feed the 256 embedded tokens through transformer layers, where self-attention mixes global and local context:

$$f'_{\text{patch}} = \text{SelfAttention}(f_{\text{patches}}),$$

letting each patch "talk" to all other patches.

Step 4. Per-patch features After the final layer, each patch becomes a fixed-dimensional feature vector. The dimensionality depends on the ViT model variant:

- ViT-G/14 → 1536-D
- ViT-L/14 → 1024-D
- ViT-B/14 → 768-D

Hence the raw output tensor has shape

$$[\text{num patches}, \text{feature dim}] = [256, 768 \text{ (or } 1024/1536)].$$

What you can do with the 256 embeddings

- Patch-to-patch matching across images or video frames: for each patch feature f^A_{patch} in image A, find its nearest feature f^B_{patch} in image B by minimizing

$$\arg\min_j \|f^A_{\text{patch}} - f^B_{\text{patch}}(j)\|_2.$$

- Fine-grained part correspondence (e.g., key-point transfer, segmentation refinement) by comparing patch embeddings.
- 2D↔3D alignment in pipelines such as FreeZe: synthetic CAD renders provide patch features for 3D query points, which are then matched to real RGB-D observations for zero-shot 6-DoF pose estimation.

In short, DINOv2 turns every 14×14 patch into a rich, context-aware descriptor that plugs seamlessly into a wide range of vision applications.

5.5 Geometric Descriptors

Geometric descriptors map the local 3D structure around a point $\mathbf{p} \in \mathbb{R}^3$ to a compact vector $\mathbf{g}_p \in \mathbb{R}^d$ such that

$$\mathbf{g}_p \approx \mathbf{g}_q \iff \text{patch}(\mathbf{p}) \sim \text{patch}(\mathbf{q}),$$

i.e., neighborhoods with similar shape yield nearby embeddings under a chosen metric. We investigate three families:

- **FPFH/SHOT (handcrafted):** histogram-based, fast, training-free;
- **FCGF (learned sparse conv):** data-driven geometric descriptors with sparse 3D CNNs;
- **GeDi (+ PointNet++) and GEDI fusion:** self-supervised, invariant geometric descriptors fused with visual features into a single 128-D vector.

These are used to build 3D–3D correspondences for pose estimation in a frozen FreeZe pipeline.

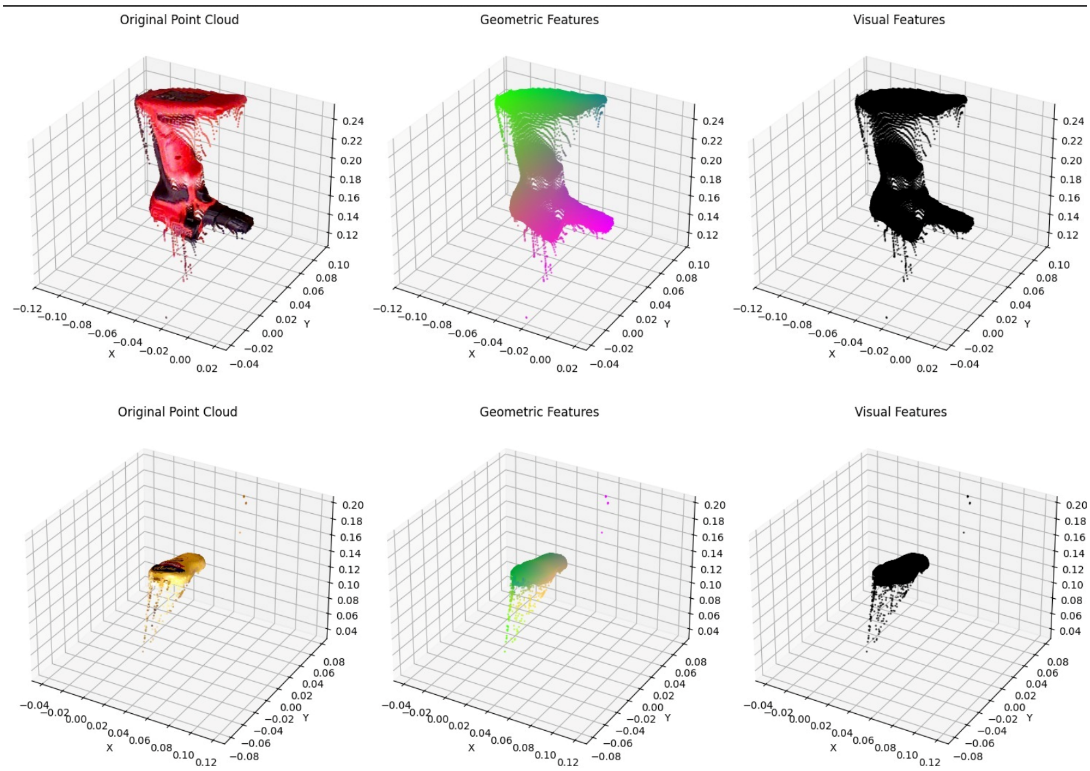


Figure 22: Implementation results on YCB objects. Rows (top: drill, bottom: mustard). Columns (left→right): raw RGB point cloud, geometric descriptors, visual descriptors. Geometry captures local shape; vision captures appearance. Their complementarity motivates our fused GEDI representation.

5.5.1 GeDi + PointNet++

For an anchor $\mathbf{p}$, define its r-ball neighborhood

$$\mathcal{N}_r(\mathbf{p}) = \{\mathbf{q}_i \mid \|\mathbf{q}_i - \mathbf{p}\|_2 \le r\}. \tag{3}$$

Local reference frame (LRF). Covariance:

$$\mathbf{C} = \frac{1}{|\mathcal{N}_r|} \sum_{\mathbf{q}_i \in \mathcal{N}_r} (\mathbf{q}_i - \mathbf{p})(\mathbf{q}_i - \mathbf{p})^\top = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top,$$

$\mathbf{V}$ provides principal axes; $\mathbf{R} = \mathbf{V}^\top$ aligns the patch (signs fixed via normals).

Canonicalisation.

$$\mathbf{q}_i' = \frac{\mathbf{R}(\mathbf{q}_i - \mathbf{p})}{\max_j \|\mathbf{R}(\mathbf{q}_j - \mathbf{p})\|_2}.$$

Permutation-invariant encoding. A shared MLP (PointNet++ style) encodes each point:

$$\mathbf{h}_i = \phi(\mathbf{q}_i') \in \mathbb{R}^{256}, \qquad \mathbf{h}_{\max} = \max_i \mathbf{h}_i,$$

then a FC layer reduces to 32-D:

$$\mathbf{d}_r = \mathrm{FC}_{256\to 32}(\mathbf{h}_{\max}), \qquad \mathbf{d}_r \leftarrow \frac{\mathbf{d}_r}{\|\mathbf{d}_r\|_2}.$$

Multi-scale fusion. For two radii r_1, r_2:

$$\mathbf{f}_{\text{geom}}^{64} = \mathbf{d}_{r_1}^{32} \,\|\, \mathbf{d}_{r_2}^{32}.$$

Training. Self-supervised contrastive loss:

$$\mathcal{L}_{\text{ctr}} = -\log \frac{\exp(\mathbf{f}_a^{+\top} \mathbf{f}_b^{+} / \tau)}{\sum_k \exp(\mathbf{f}_a^{+\top} \mathbf{f}_k^{-} / \tau)},$$

optionally plus an RPMG distance-preservation term. No task labels required.

5.5.2 FCGF (Fully Convolutional Geometric Features)

Let $\mathcal{P} = \{\mathbf{p}_i\}$. After voxelization, a sparse 3D CNN encoder applies layers

$$\mathbf{X}^{\ell+1} = \sigma\left(\mathrm{SparseConv}^{(\ell)}(\mathbf{X}^{\ell})\right),$$

downsampling resolution (1×, 2×, 4×, 8×). Features at the bottleneck are interpolated back to original points:

$$\mathbf{g}_i = \mathrm{Interp}(\mathbf{X}^L, \mathbf{p}_i), \quad \mathbf{g}_i \in \mathbb{R}^d.$$

Losses: InfoNCE or triplet margin with hard negatives. Pros: robust, learns invariances; Cons: needs supervised or large pretraining data, purely geometric.

5.5.3 FPFH / SHOT

FPFH. For point $\mathbf{p}$ (normal $\mathbf{n}_p$) and neighbor $\mathbf{q}$ (normal $\mathbf{n}_q$), define Darboux frame at $\mathbf{p}$:

$$\mathbf{u} = \mathbf{n}_p, \quad \mathbf{v} = \frac{(\mathbf{q}-\mathbf{p}) - ((\mathbf{q}-\mathbf{p})\cdot\mathbf{u})\mathbf{u}}{\|(\mathbf{q}-\mathbf{p}) - ((\mathbf{q}-\mathbf{p})\cdot\mathbf{u})\mathbf{u}\|}, \quad \mathbf{w} = \mathbf{u} \times \mathbf{v}.$$

Angular features:

$$\alpha = \mathbf{v}\cdot\mathbf{n}_q, \quad \phi = \mathbf{u}\cdot\frac{\mathbf{q}-\mathbf{p}}{\|\mathbf{q}-\mathbf{p}\|}, \quad \theta = \arctan\left(\frac{\mathbf{w}\cdot\mathbf{n}_q}{\mathbf{u}\cdot\mathbf{n}_q}\right).$$

SPFH bins (α, ϕ, θ); FPFH combines SPFHs of $\mathbf{p}$ and neighbors (33-D histogram). Fast, but sensitive to noise and lacks texture cues.

SHOT. Builds an LRF, partitions spherical support into bins (radius, azimuth, elevation); concatenates occupancies into a 134-D histogram. Still handcrafted; same limitations.

5.5.4 Comparative Analysis: FPFH/SHOT vs. FCGF vs. GEDI

Criterion	**FPFH / SHOT**	**FCGF**	**GEDI (Ours)**
Texture / appearance discrimination	None	None	Strong
Sensitivity to noise / normals	High	Medium	Low
Training requirement	None	Supervised / large pretraining	None (single W)
Ability to disambiguate similar shapes	Limited	Good	Excellent
Clutter / occlusion robustness	Medium	Good	Very high
Integration in frozen pipeline	Medium	Medium	Very high
Overall verdict	Lightweight but weak	Strong geometry-only	Best fit for FreeZe

Table 3: Comparison of geometric descriptor families for the FreeZe pipeline. The GEDI column is highlighted as the preferred choice.

FPFH/SHOT

- No appearance cues; cannot distinguish visually different but geometrically similar regions.
- Sensitive to normal noise and sparse/degraded depth.

FCGF

- Requires large annotated/pretrained datasets; domain shift can degrade performance.
- Still purely geometric, thus ambiguous for similar shapes with different textures.

GEDI

- Combines geometry and vision in a single, compact 128-D vector.
- No per-object retraining; one projection W can be frozen.
- Ideal for FreeZe: template-driven, prompt-free, robust in clutter/occlusion.

5.5.5 Conclusion

FreeZe requires zero-shot, robust correspondences under clutter, illumination changes, and sensor noise. It also demands compact descriptors for fast nearest-neighbor search. Only **GEDI** satisfies all of these simultaneously by fusing visual and geometric cues in a frozen, feed-forward pipeline. Consequently, GEDI is our recommended descriptor for FreeZe.

5.6 Evaluation of 3D Registration and Pose Refinement Strategies

5.6.1 GeoTransformer: Architecture and Loss Functions

Pipeline (coarse→fine).

1. **Backbone & Superpoints.** KPConv-FPN downsamples P, Q to superpoints $\hat{P}, \hat{Q}$ (coarsest level) and produces multi-scale features.
2. **Geometric Transformer.** Interleaved *geometric self-attention* (intra-cloud) and *feature cross-attention* (inter-cloud) yield hybrid features $H_{\hat{P}}, H_{\hat{Q}}$ for superpoint matching.
3. **Superpoint Matching.** Gaussian correlation + dual-normalization selects top-N_c patch pairs.
4. **Point Matching.** For each matched patch pair, an optimal-transport (Sinkhorn) layer gives dense correspondences.
5. **RANSAC-free Pose.** Local-to-global registration (LGR) solves weighted SVD per match and refines globally.

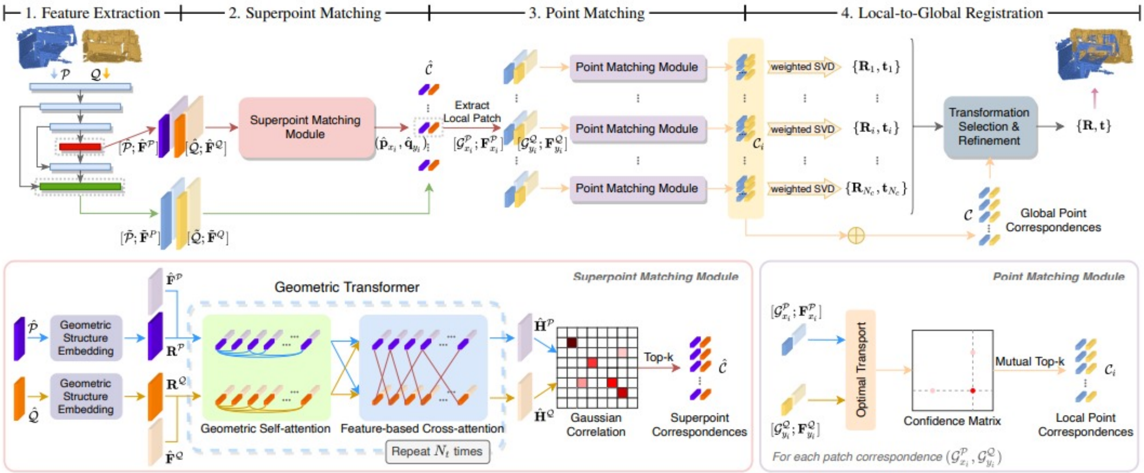

1. Feature Extraction
2. Superpoint Matching
3. Point Matching
4. Local-to-Global Registration
$\mathcal{P}$
$\mathcal{Q}$
$[\hat{\mathcal{P}}; \hat{\mathbf{F}}^{\mathcal{P}}]$
$[\hat{\mathcal{Q}}; \hat{\mathbf{F}}^{\mathcal{Q}}]$
Superpoint Matching Module
$\hat{\mathcal{C}}$
$(\hat{\mathbf{p}}_{x_i}, \hat{\mathbf{q}}_{y_i})$
Extract Local Patch
$[\mathcal{G}^{\mathcal{P}}_{x_i}; \mathbf{F}^{\mathcal{P}}_{x_i}]$
$[\mathcal{G}^{\mathcal{Q}}_{y_i}; \mathbf{F}^{\mathcal{Q}}_{y_i}]$
$[\tilde{\mathcal{P}}; \tilde{\mathbf{F}}^{\mathcal{P}}]$
$[\tilde{\mathcal{Q}}; \tilde{\mathbf{F}}^{\mathcal{Q}}]$
Point Matching Module
Point Matching Module
Point Matching Module
$\mathcal{C}_i$
weighted SVD
weighted SVD
weighted SVD
$\{\mathbf{R}_1, \mathbf{t}_1\}$
$\{\mathbf{R}_i, \mathbf{t}_i\}$
$\{\mathbf{R}_{N_c}, \mathbf{t}_{N_c}\}$
Transformation Selection & Refinement
$\{\mathbf{R}, \mathbf{t}\}$
$\mathcal{C}$
Global Point Correspondences
Superpoint Matching Module
Geometric Transformer
$\hat{\mathcal{P}}$
$\hat{\mathcal{Q}}$
Geometric Structure Embedding
Geometric Structure Embedding
$\hat{\mathbf{F}}^{\mathcal{P}}$
$\mathbf{R}^{\mathcal{P}}$
$\mathbf{R}^{\mathcal{Q}}$
$\hat{\mathbf{F}}^{\mathcal{Q}}$
Geometric Self-attention
Feature-based Cross-attention
Repeat N_t times
$\hat{\mathbf{H}}^{\mathcal{P}}$
$\hat{\mathbf{H}}^{\mathcal{Q}}$
Gaussian Correlation
Top-k
$\hat{\mathcal{C}}$
Superpoint Correspondences
Point Matching Module
$[\mathcal{G}^{\mathcal{P}}_{x_i}; \mathbf{F}^{\mathcal{P}}_{x_i}]$
$[\mathcal{G}^{\mathcal{Q}}_{y_i}; \mathbf{F}^{\mathcal{Q}}_{y_i}]$
Optimal Transport
Confidence Matrix
Mutual Top-k
$\mathcal{C}_i$
Local Point Correspondences
For each patch correspondence $(\mathcal{G}^{\mathcal{P}}_{x_i}, \mathcal{G}^{\mathcal{Q}}_{y_i})$

Figure 23: The backbone downsamples the input point clouds and learns features in multiple resolution levels. The Superpoint Matching Module extracts high-quality superpoint correspondences between $\hat{P}$ and $\hat{Q}$ using the Geometric Transformer, which iteratively encodes intra-point-cloud geometric structures and inter-point-cloud geometric consistency. The superpoint correspondences are then propagated to dense points $\tilde{P}$ and $\tilde{Q}$ by the Point Matching Module. Finally, the transformation is computed with a local-to-global registration method.

Geometric self-attention. Given input $X \in \mathbb{R}^{|\hat{P}| \times d_t}$, the output row z_i is

$$z_i = \sum_{j=1}^{|\hat{P}|} a_{i,j}\,(x_j W_V), \quad a_{i,j} = \text{softmax}_j(e_{i,j}),$$

$$e_{i,j} = \frac{(x_i W_Q)\big(x_j W_K + r_{i,j}\big)^\top}{\sqrt{d_t}}.$$

Geometric structure embedding $r_{i,j} \in \mathbb{R}^{d_t}$ is rigid-motion invariant:

$$\rho_{i,j} = \|\hat{p}_i - \hat{p}_j\|_2, \quad r^D_{i,j,2k} = \sin\Big(\frac{\rho_{i,j}/\sigma_d}{10000^{2k/d_t}}\Big),\ r^D_{i,j,2k+1} = \cos\Big(\frac{\rho_{i,j}/\sigma_d}{10000^{2k/d_t}}\Big),$$

for k-NN neighbors $\hat{p}_x \in K_i$:

$$\alpha^x_{i,j} = \angle(\Delta_{x,i}, \Delta_{j,i}), \quad r^A_{i,j,x,2\ell} = \sin\Big(\frac{\alpha^x_{i,j}/\sigma_a}{10000^{2\ell/d_t}}\Big),\ r^A_{i,j,x,2\ell+1} = \cos\Big(\frac{\alpha^x_{i,j}/\sigma_a}{10000^{2\ell/d_t}}\Big),$$

$$r_{i,j} = \sigma\Big(r^D_{i,j} W_D + \max_{x \in K_i} r^A_{i,j,x} W_A\Big) W_R,$$

with $\sigma = \text{LeakyReLU}$. (Shared W_R cuts cost.)

Feature cross-attention. For $\hat{P} \leftarrow \hat{Q}$:

$$z^P_i = \sum_j a_{i,j}(x^Q_j W_V), \quad e_{i,j} = \frac{(x^P_i W_Q)(x^Q_j W_K)^\top}{\sqrt{d_t}}.$$

Superpoint match scoring. Gaussian correlation $s_{i,j} = \exp(-\|h^P_i - h^Q_j\|_2^2)$, then dual-normalize:

$$\bar{s}_{i,j} = \frac{s_{i,j}}{\sum_k s_{i,k}} \cdot \frac{s_{i,j}}{\sum_k s_{k,j}}.$$

Top-N_c entries form $\hat{C}$.

Point matching (Sinkhorn OT). Local cost $C_i = F^P_{x_i}(F^Q_{y_i})^\top/\sqrt{\tilde{d}}$, augment with dustbins, run Sinkhorn to get $\bar{Z}_i$, drop dustbins $\Rightarrow Z_i$. Mutual top-k on Z_i yields dense set C.

RANSAC-free pose (LGR). Local weighted SVD per pair C_i:

$$(R_i, t_i) = \arg\min_{R,t} \sum_{(\tilde{p},\tilde{q}) \in C_i} w \, \|R\tilde{p} + t - \tilde{q}\|_2^2.$$

Pick global best (R, t) by max inliers, iterate N_r times with re-weighting.

Losses. Overlap-aware circle loss (superpoints): For anchor patch G^P_i, positives ε^p_i, negatives ε^n_i,

$$\mathcal{L}^P_{\text{oc}} = \frac{1}{|A|} \sum_{G^P_i \in A} \log\Big[1 + \Big(\sum_{j \in \varepsilon^p_i} e^{\lambda^j_i \beta^p_{i,j}(d^j_i - \Delta_p)}\Big)\Big(\sum_{k \in \varepsilon^n_i} e^{\beta^n_{i,k}(\Delta_n - d^k_i)}\Big)\Big],$$

$d^j_i = \|h^P_i - h^Q_j\|_2$, $\lambda^j_i = \sqrt{o^j_i}$ (patch-overlap weight), $\beta^p_{i,j} = \gamma(d^j_i - \Delta_p)$, $\beta^n_{i,k} = \gamma(\Delta_n - d^k_i)$. $\mathcal{L}_{\text{oc}} = (\mathcal{L}^P_{\text{oc}} + \mathcal{L}^Q_{\text{oc}})/2$.

Point matching loss (OT): For each ground-truth patch match i,

$$\mathcal{L}_{p,i} = -\sum_{(x,y) \in M_i} \log \bar{z}^i_{x,y} - \sum_{x \in I_i} \log \bar{z}^i_{x,m_i+1} - \sum_{y \in J_i} \log \bar{z}^i_{n_i+1,y}, \quad \mathcal{L}_p = \frac{1}{N_g} \sum_i \mathcal{L}_{p,i}.$$

Total loss: $\mathcal{L} = \mathcal{L}_{\text{oc}} + \mathcal{L}_p$.

5.6.2 CoFiNet: Confidence-Based Correspondence Filtering

CoFiNet is a lightweight and effective model designed to filter out incorrect correspondences between point clouds. It enhances the quality of initially estimated correspondences by learning to predict per-correspondence confidence scores based on local geometric structure. CoFiNet is particularly effective when the coarse matching stage produces noisy or partially correct alignments.

Architecture:

- **Graph Construction:** For each candidate correspondence (p_i, q_i) between the source and target point clouds, CoFiNet constructs a local graph G_i consisting of the k-nearest neighbors in both source and target neighborhoods. These local graphs encode contextual geometry and are the basis for further aggregation.
- **Positional Encoding:** Each edge in the local graph is embedded using a learned positional encoding function $\phi(p_i - p_j)$ to model the relative position between nodes.
- **Feature Aggregation:** A series of transformer layers are applied to these graphs. Self-attention is used to capture intra-cloud geometry, while cross-attention is used to incorporate inter-cloud context, refining the correspondence descriptors.
- **Confidence Prediction:** A Multi-Layer Perceptron (MLP) takes the refined descriptor and outputs a confidence score $s_i \in [0, 1]$ indicating the likelihood that (p_i, q_i) is a correct correspondence.

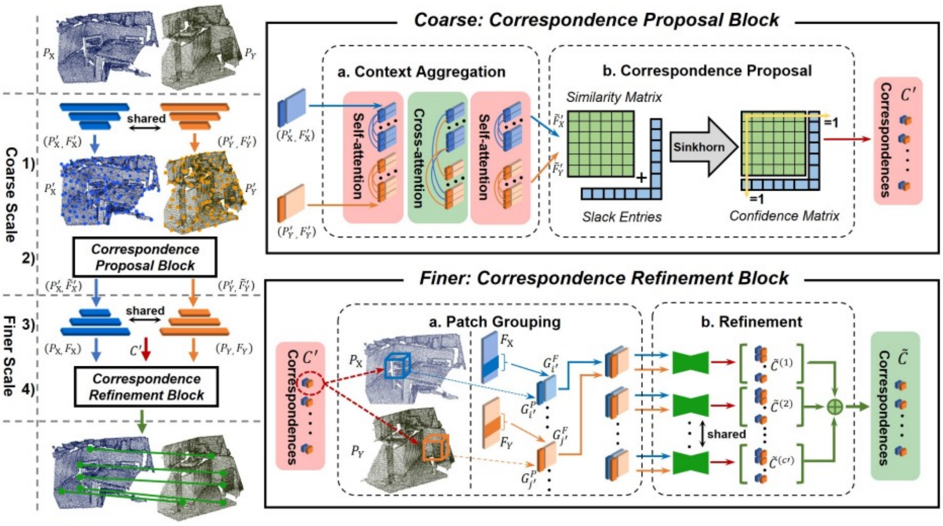

Coarse: Correspondence Proposal Block
a. Context Aggregation
b. Correspondence Proposal
Self-attention
Cross-attention
Self-attention
Similarity Matrix
Slack Entries
Sinkhorn
Confidence Matrix
=1
=1
Correspondences
C'
(P'_X, F'_X)
(P'_Y, F'_Y)
$\tilde{F}'_X$
$\tilde{F}'_Y$
P_X
P_Y
shared
Coarse Scale
1)
2)
P'_X
P'_Y
Correspondence Proposal Block
$(P'_X, \tilde{F}'_X)$
$(P'_Y, \tilde{F}'_Y)$
Finer Scale
3)
4)
(P_X, F_X)
(P_Y, F_Y)
Correspondence Refinement Block
Finer: Correspondence Refinement Block
a. Patch Grouping
b. Refinement
F_X
F_Y
$G^F_{i'}$
$G^P_{i'}$
$G^F_{j'}$
$G^P_{j'}$
$\tilde{C}^{(1)}$
$\tilde{C}^{(2)}$
$\tilde{C}^{(c')}$
$\tilde{C}$

Figure 24: Overview of CoFiNet. From top to bottom: (1) Dense points are downsampled to uniformly distributed nodes with shared feature learning. (2) The Correspondence Proposal Block aggregates contextual information using self- and cross-attention and computes a similarity matrix, followed by Sinkhorn normalization to generate coarse correspondences. (3) Features are decoded to dense descriptors. (4) The Correspondence Refinement Block groups patches and refines them to point-level correspondences using the density-adaptive matching module.

Loss Functions:

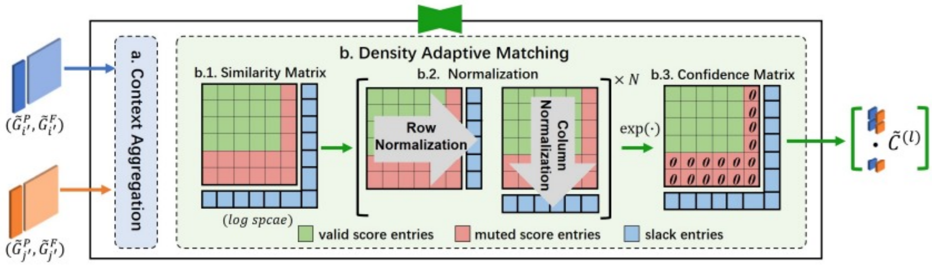

$(\tilde{G}^P_{i'}, \tilde{G}^F_{i'})$
$(\tilde{G}^P_{j'}, \tilde{G}^F_{j'})$
a. Context Aggregation
b. Density Adaptive Matching
b.1. Similarity Matrix
b.2. Normalization
Row Normalization
Column Normalization
$\times N$
$\exp(\cdot)$
b.3. Confidence Matrix
0
(log spcae)
🟩 valid score entries
🟥 muted score entries
🟦 slack entries
$\tilde{C}^{(l)}$

Figure 25: Illustration of the Density-Adaptive Matching Module in CoFiNet. (a) Context aggregation is performed using the same module as in the proposal block. (b.1) A similarity matrix is computed, with slack entries initialized to 0 and muted entries set to $-\infty$. The Sinkhorn algorithm is run with row and column normalization over N iterations. (b.3) The final confidence matrix is formed and high-confidence correspondences are selected for refinement.

- **Binary Cross-Entropy for Confidence:**

  $$\mathcal{L}_{\text{conf}} = -\sum_i \left[y_i \log s_i + (1 - y_i) \log(1 - s_i) \right], \tag{4}$$

  where y_i is a binary indicator of whether correspondence i is valid.

- **Weighted Pose Regression:**

  $$\mathcal{L}_{\text{pose}} = \sum_i s_i \cdot \|T(p_i) - q_i\|^2, \tag{5}$$

  where high-confidence matches contribute more to the final transformation loss.

Training Details: CoFiNet is trained using ground-truth pose annotations and clean, labeled correspondences. The objective is to learn a filtering function that down-weights or eliminates noisy matches from an external correspondence estimator (e.g., FCGF or SuperGlue).

Inference Pipeline: Given a set of initial putative correspondences, CoFiNet processes each through its graph-based transformer module and outputs scores s_i. A threshold is applied to discard low-confidence matches. The remaining subset is passed to a classical solver such as weighted SVD or RANSAC for pose estimation.

Limitations in Freeze Context:

- **Requires Initial Matches:** CoFiNet is not a standalone matcher—it requires coarse input correspondences from a separate method.
- **Supervised Dependence:** Its performance relies heavily on training with high-quality labeled inlier/outlier correspondence datasets, which are not available in CAD-based zero-shot settings.
- **Domain Generalization Gap:** Since Freeze operates on 3D renderings of CAD models and RGB images with no domain overlap during training, the statistical assumptions of CoFiNet's training regime do not hold.
- **Not Fully Frozen:** CoFiNet does not conform to the Freeze philosophy of using frozen foundation models—it needs to be trained and tuned per dataset.

Conclusion: While CoFiNet is elegant and lightweight, its reliance on external matchers, labeled data, and supervised learning pipelines makes it unsuitable for integration into the Freeze pipeline. Freeze benefits more from unsupervised, deterministic methods like RANSAC, SAR, and ICP that do not depend on training distribution alignment.

5.6.3 OverlapPredator: Predicting Overlapping Regions Between Point Clouds

OverlapPredator predicts overlapping visibility masks and high-confidence matches for partial point cloud registration. It explicitly models overlap and confidence via multi-head networks.

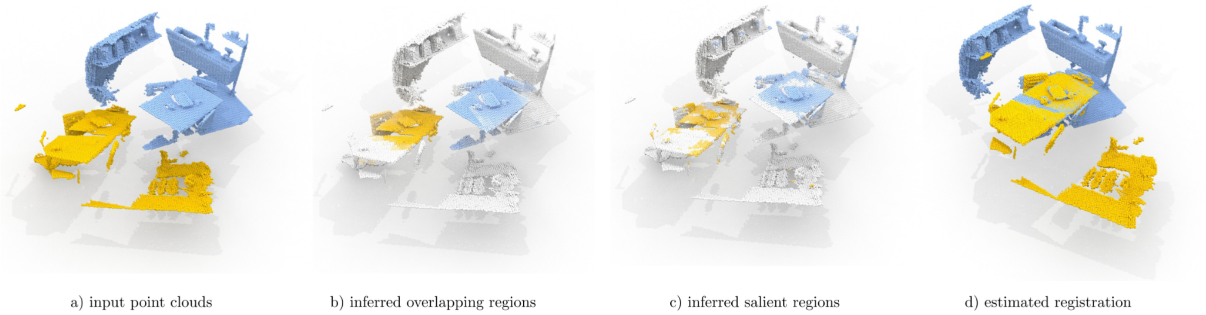

a) input point clouds
b) inferred overlapping regions
c) inferred salient regions
d) estimated registration

Figure 26: Visualization of Overlap Predator's registration pipeline. (a) Input point clouds from two partially overlapping scans, colored in yellow and blue. (b) Predicted overlapping regions where the model infers areas likely to correspond between the two point clouds. (c) Predicted salient regions highlighting geometrically distinctive parts most useful for matching. (d) Final estimated registration showing the aligned point clouds after establishing robust correspondences. OverlapPredator uses overlap and saliency predictions to guide point cloud registration, achieving reliable alignment even under partial overlap or challenging scenes.

Architecture:

- **Feature Extraction:** Shared ResUNet for pointwise features.
- **Coarse Matching:** Cosine similarity between features.
- **Overlap Prediction Head:** $o_i = \sigma(h(f_i))$ predicts visibility.
- **Inlier Prediction Head:** Confidence estimation c_i for match validity.

Loss Functions:

- **Overlap BCE Loss:**

$$\mathcal{L}_{\text{overlap}} = -\sum_i [o_i^* \log o_i + (1 - o_i^*) \log(1 - o_i)] \tag{6}$$

- **Inlier Classification Loss (Focal):**
- **Weighted Pose Regression Loss:**

Discussion: OverlapPredator is ideal for real-world scans with clutter and occlusion, but it is computationally expensive and dependent on real-world scan-specific priors. Freeze, by contrast, operates on clean CAD models where such overlap reasoning is unnecessary.

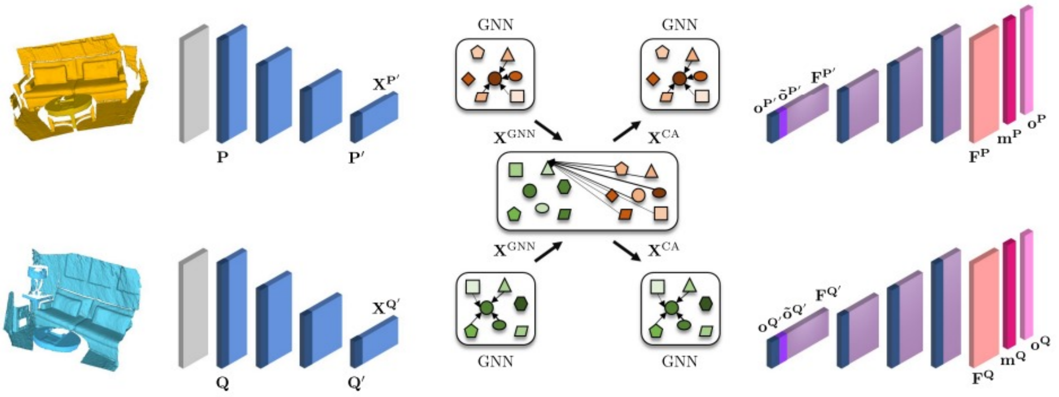

GNN
GNN
$X^{P'}$
P
P'
X^{GNN}
X^{CA}
$o^{P'}, \tilde{o}^{P'}$
$F^{P'}$
F^{P}
m^{P}
o^{P}
X^{GNN}
X^{CA}
$X^{Q'}$
Q
Q'
GNN
GNN
$o^{Q'}, \tilde{o}^{Q'}$
$F^{Q'}$
F^{Q}
m^{Q}
o^{Q}

Figure 27: **PREDATOR pipeline.** Voxelized point clouds $\mathbf{P}, \mathbf{Q}$ are encoded to superpoints $\mathbf{P}_0, \mathbf{Q}_0$ with latent features $\mathbf{X}_{\mathbf{P}_0}, \mathbf{X}_{\mathbf{Q}_0}$. A stack of self-attention (GNN) and cross-attention (CA) blocks injects co-contextual information and predicts overlap scores $o_{\mathbf{P}_0}, o_{\mathbf{Q}_0}$ and cross-overlap scores $\tilde{o}_{\mathbf{P}_0}, \tilde{o}_{\mathbf{Q}_0}$. The decoder upsamples these conditioned features to per-point descriptors $\mathbf{F}_{\mathbf{P}}, \mathbf{F}_{\mathbf{Q}}$, per-point overlap $o_{\mathbf{P}}, o_{\mathbf{Q}}$, and matchability $m_{\mathbf{P}}, m_{\mathbf{Q}}$.

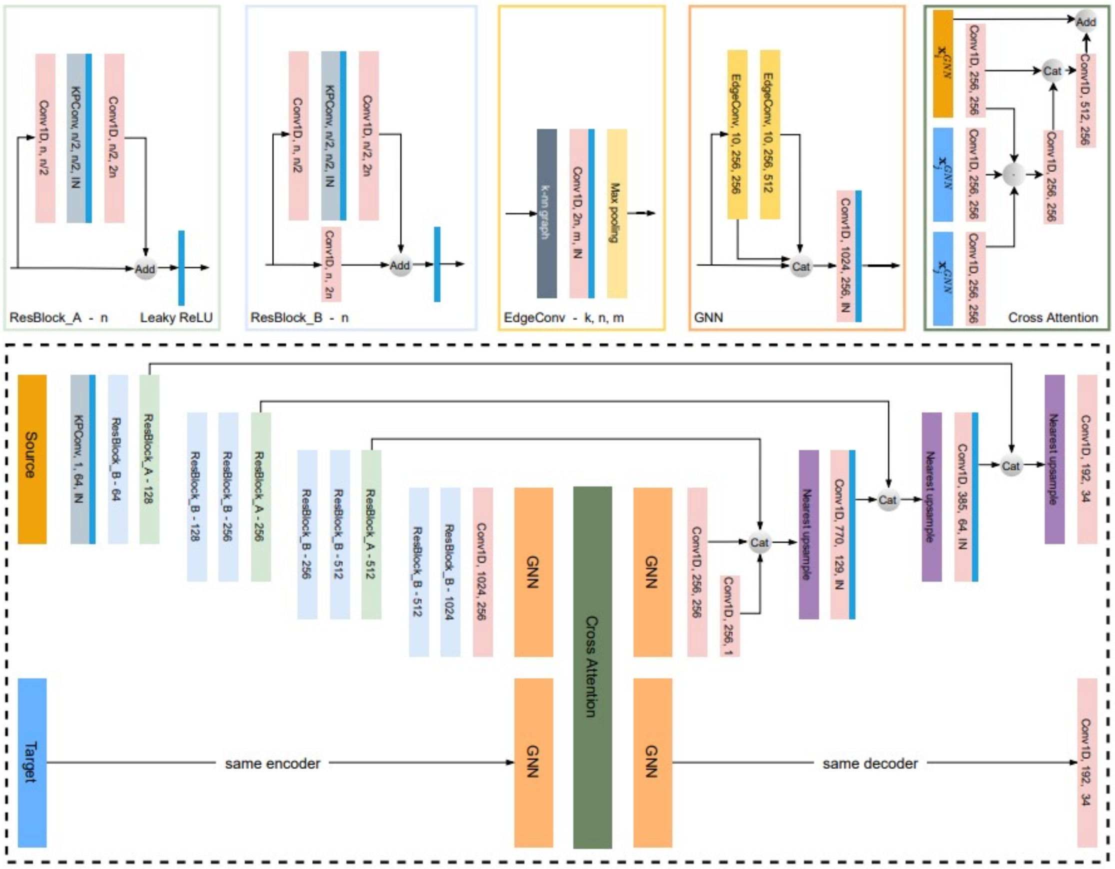

Conv1D, n, n/2
KPConv, n/2, n/2, IN
Conv1D, n/2, 2n
Add
ResBlock_A - n
Leaky ReLU
Conv1D, n, n/2
KPConv, n/2, n/2, IN
Conv1D, n/2, 2n
Conv1D, n, 2n
Add
ResBlock_B - n
k-nn graph
Conv 1D, 2n, m, IN
Max pooling
EdgeConv - k, n, m
EdgeConv, 10, 256, 256
EdgeConv, 10, 256, 512
Conv1D, 1024, 256, IN
Cat
GNN
x_i^{GNN}
x_j^{GNN}
x_j^{GNN}
Conv 1D, 256, 256
Conv 1D, 256, 256
Conv 1D, 256, 256
Cat
Add
Conv1D, 256, 256
Conv1D, 512, 256
Cross Attention
Source
KPConv, 1, 64, IN
ResBlock_B - 64
ResBlock_A - 128
ResBlock_B - 128
ResBlock_B - 256
ResBlock_A - 256
ResBlock_B - 256
ResBlock_B - 512
ResBlock_A - 512
ResBlock_B - 512
ResBlock_B - 1024
Conv1D, 1024, 256
GNN
Cross Attention
GNN
Conv1D, 256, 256
Conv1D, 256, 1
Cat
Nearest upsample
Conv1D, 770, 129, IN
Cat
Nearest upsample
Conv1D, 385, 64, IN
Cat
Nearest upsample
Conv1D, 192, 34
Target
same encoder
GNN
GNN
same decoder
Conv1D, 192, 34

Figure 28: **Cross-attention block in PREDATOR (3DMatch setting).** For each correspondence, query $\mathbf{s}_i \in \mathbb{R}^{b\times 1}$, key $\mathbf{k}_i \in \mathbb{R}^{b\times 1}$ and value $\mathbf{v}_i \in \mathbb{R}^{b\times 1}$ are reshaped into 4 heads of size $b/4$. Scores are computed as $\mathbf{S} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)$, and messages are aggregated as $\mathbf{M} = \mathbf{S}\mathbf{V}$, then concatenated and projected back to b channels.

5.6.4 Classical vs. Learned Pose-Refinement Methods

Table 4 contrasts our **RANSAC → ICP → SAR** stack with recent learned registration networks. A complete, illustrated description of the RANSAC → ICP → Symmetry-Aware Refinement pipeline is provided in Sec. 8.

Criterion	**RANSAC + ICP + SAR**	**GeoTransformer / CoFiNet / Overlap-Predator**
Training requirement	Training-free	Supervised training required
Domain generalization	Robust to domain shift	Often needs re-training
Modularity / interpretability	Modular, fully interpretable	Monolithic, opaque
Symmetry handling	Native via SAR	Not supported natively
Real-time deployment	Fast	Often slow
External matcher required	No	Yes (additional filtering)
Scalability	High	Limited

Table 4: Comparative evaluation of classical refinement versus modern learned approaches.

6 Query Pipeline

For every CAD object, we construct a dataset

$$\mathcal{D} = \{(\mathbf{p}_i,\, \mathbf{f}_i)\}_{i=1}^{N}$$

with $N = 5{,}000$ surface points $\mathbf{p}_i \in \mathbb{R}^3$ and fused 128-dimensional descriptors $\mathbf{f}_i \in \mathbb{S}^{127}$ (the unit hypersphere in $\mathbb{R}^{128}$).

6.1 CNOS Multi-View Rendering

View set. An icosahedral sphere of radius $R = 1\,\mathrm{m}$ is subdivided once, yielding $K = 42$ regularly spaced camera directions:

$$C_k = R_z(\psi_k)\frac{\widehat{\mathbf{v}}_k}{\|\widehat{\mathbf{v}}_k\|_2} \in \mathrm{SO}(3), \qquad \psi_k \sim \mathcal{U}(0, 2\pi)$$

where $R_z(\cdot)$ denotes a random rotation about the z-axis, and $\widehat{\mathbf{v}}_k$ are icosahedral directions. The worst-case angular gap in coverage is

$$\theta_{\max} \leq \arccos\left(1 - \frac{4\pi}{5\sqrt{5}K}\right) \approx 16.4^\circ$$

ensuring nearly uniform multi-view coverage.

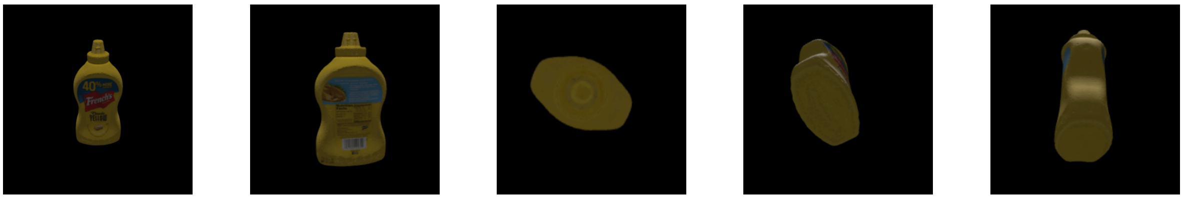


Figure 29: Five example images out of the 42 generated using icosahedron sampling in CNOS synthetic rendering. These multi-view renders cover different angles of the object for robust feature extraction.

Each render uses fixed intrinsics

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Why 42 Renderings?

We place 42 cameras evenly around the object on a sphere of radius 1 meter, using points from a subdivided icosahedron to ensure uniform coverage. This setup guarantees that from any viewing angle, there is a camera within about 16.4°. Such wide and balanced coverage ensures the object is well captured from all directions, supporting robust feature extraction and recognition.

6.2 Indexed Point Cloud ($N_q = 5000$)

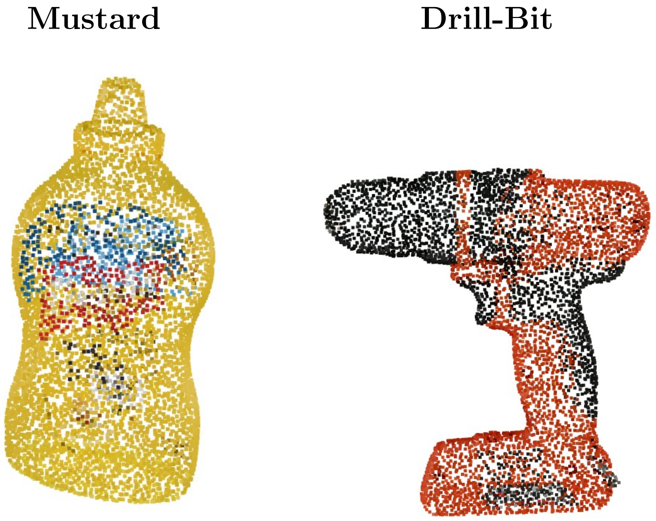


Figure 30: Texture-coloured 5k-point clouds.

We uniformly sample $N = 5{,}000$ vertices $\{\mathbf{p}_i\}$ on the CAD mesh, extract and attach each point's RGB texture, and apply a global scaling factor 10^{-3} for numerical conditioning:

$$\mathbf{p}_i \leftarrow 10^{-3}\mathbf{p}_i$$

The integer index i is kept constant across all downstream stages, ensuring full correspondence between appearance and geometry at each point.

Why Uniform Sampling of 5,000 Points?
Uniformly sampling ensures consistent and evenly distributed coverage of the surface, making sure no detail is missed and that descriptors computed later correspond meaningfully to fixed points on the object. Using exactly 5,000 points is a balance between resolution and computational efficiency.

6.3 Visual Descriptor

For each visible point–view pair $(\mathbf{p}_i, I_k)$, a DINOv2 backbone yields a 1536-dimensional token. These are mean-pooled over all visible views to obtain a per-point embedding, then dimensionally reduced:

$$v_n^Q = \text{PCA-Reduce}\left(\frac{1}{|\mathcal{V}_n|}\sum_{k\in\mathcal{V}_n}\mathbf{t}_{nk}\right), \quad v_n^Q \in \mathbb{R}^{64}$$

where $\mathcal{V}_n$ is the set of views where nth point is visible, and $\mathbf{t}_{nk}$ is the DINOv2 token for that pair.

Principal component analysis (PCA) reduces these to 64 dimensions, preserving over 96% of variance.

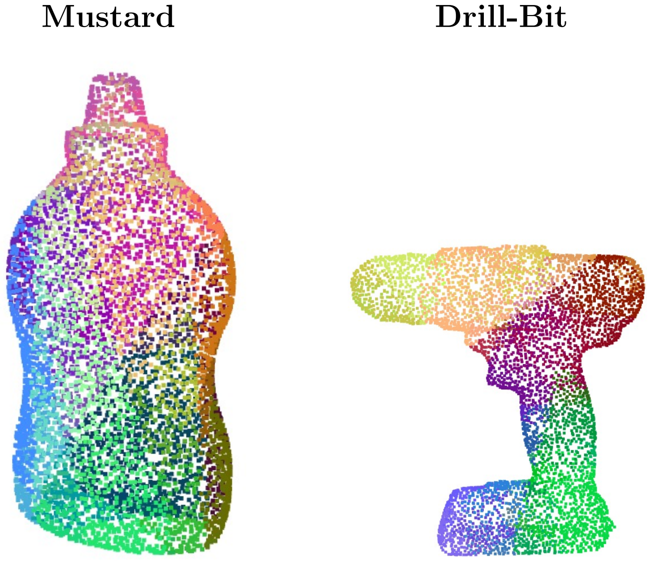


Figure 31: 64-D visual descriptors (RGB = first three PCA axes).

Why PCA Reduction of DINOv2 Features?
DINOv2 outputs high-dimensional features (1536-D) that contain rich information, but such large vectors are costly to store and process. PCA reduces dimensionality to 64 while preserving most important information (96% variance), like summarizing a large essay into its main points. This speeds up computations, reduces memory, and keeps the descriptors discriminative enough for matching.

6.4 Geometry Descriptor

Geometric descriptors are computed at two scales, radii $r_1 = 0.3D$ and $r_2 = 0.4D$, where D is the mesh's bounding sphere diameter:

$$g_n^Q = \frac{[\texttt{GeDi}_{r_1}(\mathbf{p}_n) \,\|\, \texttt{GeDi}_{r_2}(\mathbf{p}_n)]}{\|[\texttt{GeDi}_{r_1}(\mathbf{p}_n) \,\|\, \texttt{GeDi}_{r_2}(\mathbf{p}_n)]\|_2}, \quad g_n^Q \in \mathbb{R}^{64}$$

where $[\cdot \,\|\, \cdot]$ denotes concatenation, and $\texttt{GeDi}_r(\mathbf{p}_n)$ computes geometric features at radius r around point $\mathbf{p}_n$.

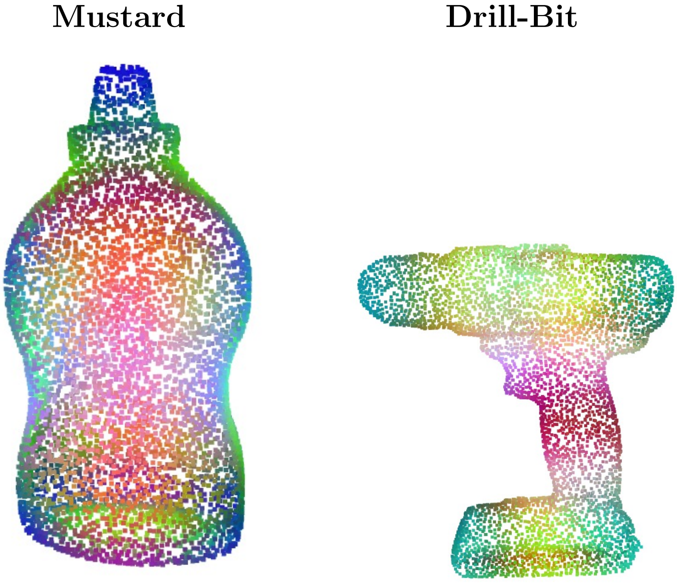


Figure 32: 64-D geometric descriptors, two radii.

Why Compute Geometry Descriptors at 0.3 and 0.4 Radii?
These two neighborhood sizes let us capture different levels of shape information around each point. The smaller radius (0.3 × diameter) captures fine local details (like textures or small features), while the larger radius (0.4 × diameter) captures broader context (overall shape and surrounding structure). This multi-scale approach improves descriptor robustness and makes the model sensitive both to detailed and global shape patterns

6.5 Fused Features

For the CAD model:

- We have 5000 points on the CAD model.
- For each point n, we extracted:
  - A visual feature vector $v_n^Q \in \mathbb{R}^{64}$, reduced from DINOv2,
  - A geometric feature vector $g_n^Q \in \mathbb{R}^{64}$.
- We normalized each feature vector:

$$\|v_n^Q\|_2 = \sqrt{\sum_{i=1}^{64} (v_{n,i}^Q)^2}, \quad \|g_n^Q\|_2 = \sqrt{\sum_{i=1}^{64} (g_{n,i}^Q)^2},$$

giving normalized vectors:

$$\tilde{v}_n^Q = \frac{v_n^Q}{\|v_n^Q\|_2}, \quad \tilde{g}_n^Q = \frac{g_n^Q}{\|g_n^Q\|_2}.$$

- We concatenated them into a fused feature vector:

$$f_n^Q = [\tilde{v}_n^Q, \tilde{g}_n^Q],$$

forming the full set:

$$F^Q = \{f_n^Q\}_{n=1}^{5000}.$$

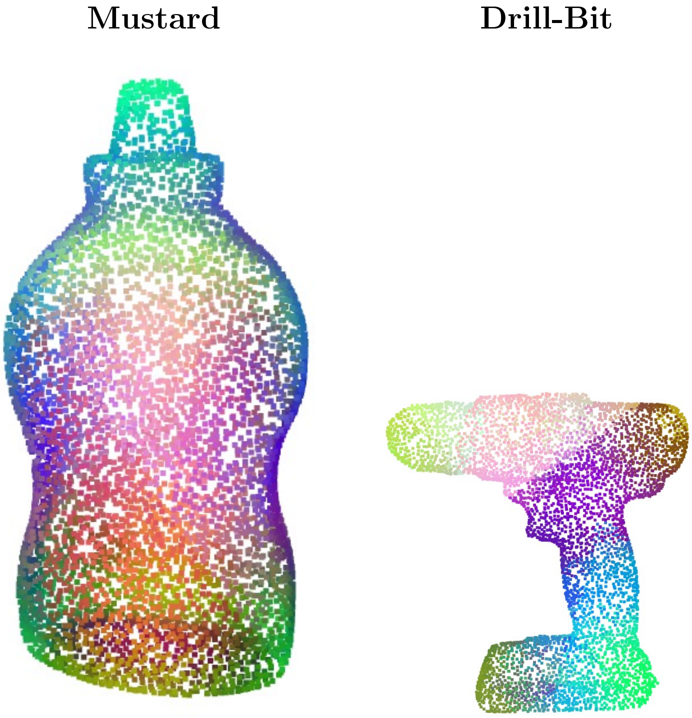


Figure 33: 128-D fused descriptors (colour = first three PCA axes).

Why Fuse Visual and Geometric Features?
Visual embeddings capture appearance, while geometric embeddings capture 3D shape context. Combining them gives a richer, more descriptive representation for each point, improving robustness to changes in viewpoint, lighting, or occlusions. This fusion also helps distinguish objects that might look similar in shape or color alone but differ when both modalities are considered together.

6.6 Summary

- **Uniform 42-view coverage:** Ensures consistent multi-view appearance, limiting maximum view drift to $\theta_{\max} \leq 16.4°$.
- **Single 5k-point index:** Perfectly aligns appearance (DINOv2) and geometry (GeDi) by design.

- **Dual-radius GeDi:** $(0.3D,\, 0.4D)$ provides a principled tradeoff between local detail and large-scale context with compact descriptors.
- **Unit-sphere descriptors:** All descriptor vectors are ℓ_2-normalized ($\|\mathbf{f}_i\|_2 = \sqrt{2}$), so geometry and appearance have equal magnitude.
- **PCA reduction:** Keeps descriptor dimension manageable, ensuring fast and memory-efficient computation while preserving essential information.
- **Fused features:** Combining visual and geometric information improves robustness and discrimination.

7 Target Pipeline

Let $I \in \mathbb{R}^{H\times W\times 3}$ be an RGB image frame, and $\mathbf{K} \in \mathbb{R}^{3\times 3}$ the camera intrinsic matrix at run-time.

7.1 Prompt Generation (Offline)

Render views $\{\tilde{I}_k\}$ of the object are processed by Florence-2 PromptGen v1.5 to generate candidate captions (text descriptions). The most frequent caption is chosen:

$$\ell_o = \arg\max_{\ell} \#\{\tilde{I}_k : \mathrm{PromptGen}(\tilde{I}_k) = \ell\}.$$

Rationale: This enables zero-shot or text-driven object detection by supplying a reliable natural language label for each target.

7.2 Instance Segmentation

NIDS-Net infers segmentation logits $\mathcal{S} \in [0,1]^{H\times W\times N}$. Each pixel (x,y) is assigned to the object j with the maximum predicted probability, constructing a binary mask:

$$M_j(x,y) = \mathbb{I}\left[\mathcal{S}_{xyj} = \max_k \mathcal{S}_{xyk}\right]$$

where $\mathbb{I}[\cdot]$ is the indicator function (1 if the condition is true, 0 otherwise).

Each binary mask $M_j(x,y)$ is defined as

$$M_j(x,y) = \begin{cases} 1 & \text{if pixel } (x,y) \text{ belongs to object } j, \\ 0 & \text{otherwise.} \end{cases}$$

Rationale: Identifying precise pixel regions belonging to each object is crucial for downstream extraction of 3D structure specific to the target.

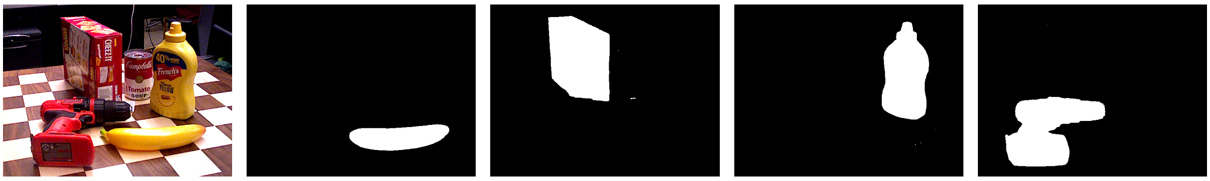


Figure 34: **Binary Mask Generation**

7.3 Depth Estimation

Using Depth-Anything, depth maps D_j are estimated across regions of interest Ω_j. Each valid pixel is back-projected to produce 3D points in the camera frame:

$$\mathbf{p}_j(x,y) = D_j(x,y)\,\mathbf{K}^{-1}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (x,y) \in \Omega_j.$$

Why back-project? This gives accurate, dense 3D points corresponding to the segmented object, essential for pose and geometry alignment.

(a) **RGB Image** (b) **Estimated Depth**

Figure 35: **Our Combined Segmentation and Depth Estimation.** We implemented a pipeline that fuses 2D masks with estimated depth, enabling 3D point clouds from single RGB images for accurate zero-shot 6D pose estimation.

7.4 Indexed Point Cloud ($N_t = 1000$)

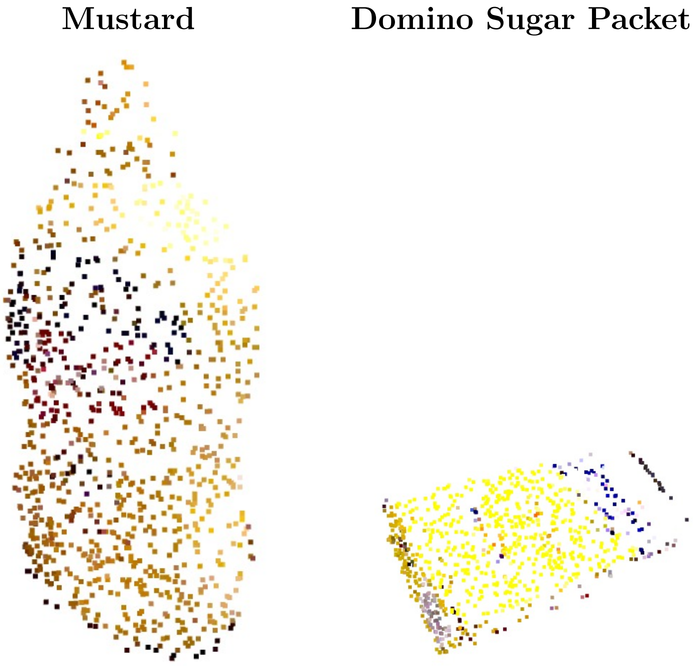


Figure 36: Texture-coloured 1k-point clouds.

From all segmented 3D points, a random subset of 1000 is selected:

$$\mathcal{I} \subset \{1, \dots, |\Omega_j|\}, \quad |\mathcal{I}| = 1000.$$

The resulting set $\{\mathbf{p}_i\}_{i\in\mathcal{I}}$ is saved. *Rationale:* Subsampling to a fixed number balances detail and computational efficiency, ensuring uniform processing during feature extraction and matching.

7.5 Visual Descriptor

The relevant crop is divided into 196 patches (14 × 14) and tokenized using DINOv2, producing $\{\mathbf{t}_q\}_{q=1}^{196} \subset \mathbb{R}^{1536}$. Each point is assigned a patch based on its image coordinates:

$$q = \lfloor u/16 \rfloor + 14\lfloor v/16 \rfloor$$

Principal component analysis (PCA) reduces each patch's descriptor to a normalized 64-dimensional vector:

$$\mathbf{v}_q = \frac{W\mathbf{t}_q}{\|W\mathbf{t}_q\|_2}, \quad \mathbf{f}_i^{\text{vis}} = \mathbf{v}_q, \quad \|\mathbf{f}_i^{\text{vis}}\|_2 = 1$$

Why PCA? Dimensionality reduction preserves most important information (variance) while making computation manageable.

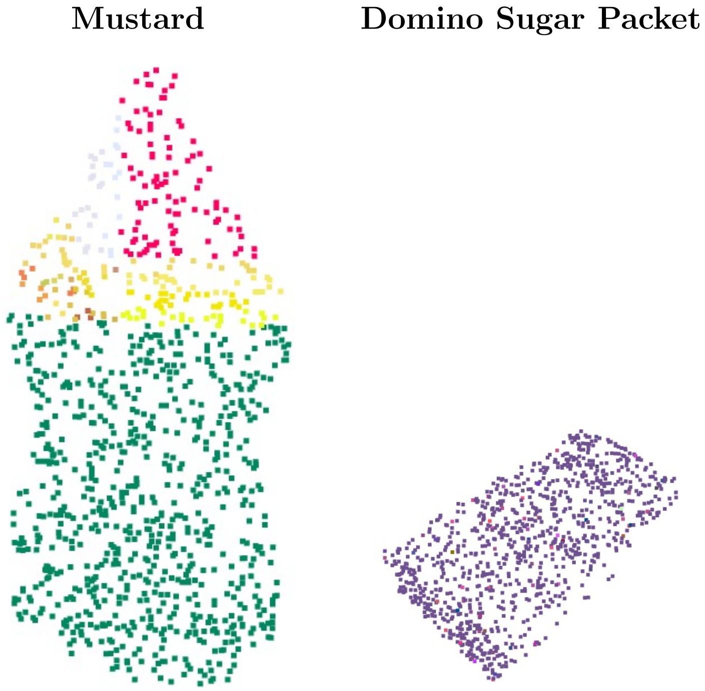


Figure 37: 64-D visual descriptors (RGB = first three PCA axes)

7.6 Geometric Descriptor

For each point $\mathbf{p}_i$, GeDi is applied at two scales ($r_1 = 0.3D_{\text{obj}}, r_2 = 0.4D_{\text{obj}}$), producing two 32-dimensional vectors. Concatenate and normalize to get:

$$\mathbf{f}_i^{\text{geo}} = \frac{\mathbf{g}_{r_1}(\mathbf{p}_i) \,\|\, \mathbf{g}_{r_2}(\mathbf{p}_i)}{\|\mathbf{g}_{r_1}(\mathbf{p}_i) \,\|\, \mathbf{g}_{r_2}(\mathbf{p}_i)\|_2}, \quad \|\mathbf{f}_i^{\text{geo}}\|_2 = 1$$

Why two radii? Multi-scale descriptors capture both fine-grained and broad geometric context, improving robustness to noise and partial views.

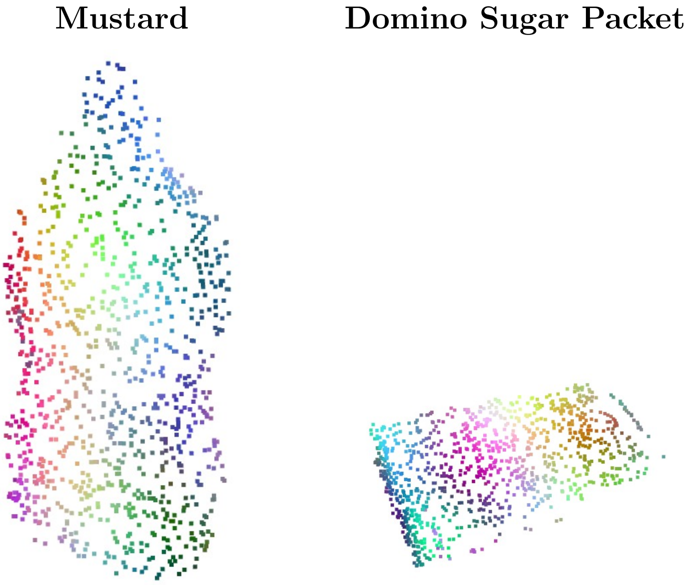


Figure 38: 64-D geometric descriptors, two radii

7.7 Fused Features

For the RGB image:

- We have 1000 points on the image.
- For each point n, we extract:
    - A visual feature vector $v_n^Q \in \mathbb{R}^{64}$, reduced from DINOv2,
    - A geometric feature vector $g_n^Q \in \mathbb{R}^{64}$.
- We normalize each feature vector:

$$\|v_n^Q\|_2 = \sqrt{\sum_{i=1}^{64} (v_{n,i}^Q)^2}, \quad \|g_n^Q\|_2 = \sqrt{\sum_{i=1}^{64} (g_{n,i}^Q)^2},$$

  obtaining normalized vectors:

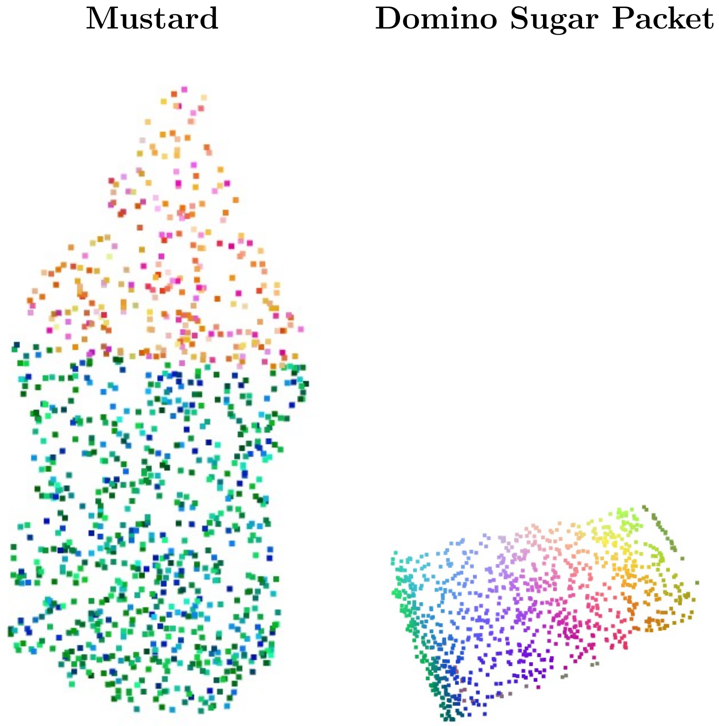
$$\tilde{v}_n^Q = \frac{v_n^Q}{\|v_n^Q\|_2}, \quad \tilde{g}_n^Q = \frac{g_n^Q}{\|g_n^Q\|_2}.$$

- We concatenate them into a fused feature vector:

$$f_n^Q = [\tilde{v}_n^Q, \tilde{g}_n^Q] \in \mathbb{R}^{128},$$

  forming the set:

$$F^Q = \{f_n^Q\}_{n=1}^{1000}.$$



Mustard

Domino Sugar Packet

Figure 39: 128-D fused descriptors (colour = first three PCA axes).

7.8 Summary

- **Automated prompt generation** produces text labels for zero-shot or open-vocabulary recognition.
- **Instance segmentation** isolates object-specific pixels, enabling precise geometry extraction.
- **Depth back-projection** allows construction of an accurate, object-centered point cloud.
- **Uniform sampling** ensures all objects are processed with equal efficiency and resolution.
- **Patchwise visual descriptors** leverage foundation models and are compressed for speed and memory savings.
- **Geometric descriptors at two scales** robustly encode local and global shape patterns, mimicking the band-pass strategy used in the query pipeline.
- **Feature fusion** (appearance + shape) produces a balanced 128-dimensional descriptor for fast, robust matching.
- **Cosine similarity** enables efficient, scalable comparison to any query descriptor, supporting reliable recognition and registration.

8 Pose estimation Pipeline

8.1 RANSAC + ICP + SVD + Symmetry-Aware Refinement (SAR)

8.1.1 RANSAC

What is RANSAC?

RANSAC (Random Sample Consensus) is a robust algorithm used to estimate model parameters (e.g., a transformation) from data that contains a significant percentage of outliers. It is especially useful in computer vision tasks such as pose estimation, where noisy or incorrect feature correspondences can negatively affect accuracy.

In our application, RANSAC is used to estimate the rigid transformation that aligns a 3D CAD model to a scene point cloud using a subset of reliable correspondences.

Context: Given a noisy set of correspondences between the CAD model and the observed scene, we want to compute a transformation $T \in SE(3)$ (the special Euclidean group in 3D) that maps model coordinates to camera (scene) coordinates.

Mathematical Formulation

Let:

- $\mathcal{C} = \{(m, n)\}$ be a set of 3D point correspondences between the scene and the model.
- $\mathbf{p}_n^Q$ be the n-th point on the CAD model.
- $\mathbf{p}_m^T$ be the corresponding point in the scene point cloud.
- $T_{\text{RANSAC}} = [R \,|\, \mathbf{t}] \in SE(3)$ be the transformation we wish to estimate.

The goal of RANSAC is to find T_{RANSAC} that maximizes the number of *inlier* correspondences, defined as:

$$\text{Inliers} = \left\{ (m, n) \in \mathcal{C} \;\middle|\; \left\| R \cdot \mathbf{p}_n^Q + \mathbf{t} - \mathbf{p}_m^T \right\| < \delta \right\}$$

Where:

- R is a 3×3 rotation matrix.
- $\mathbf{t}$ is a 3×1 translation vector.
- δ is the inlier threshold (e.g., 0.02 meters).

RANSAC works by:

1. Randomly selecting a minimal subset of k correspondences (e.g., $k = 4$).
2. Estimating a transformation T from these.
3. Counting how many of the full correspondences are inliers under this T.
4. Repeating the above steps for many iterations.
5. Returning the T with the highest inlier count.

Transformation Matrix Estimated via RANSAC

The output of RANSAC is a rigid transformation matrix T_{RANSAC} that maps 3D model points to the camera frame. It is represented as a 4×4 homogeneous matrix combining rotation and translation:

$$T_{\text{RANSAC}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This matrix allows transforming a 3D point from the model coordinate system (X_m, Y_m, Z_m) to the camera coordinate system (X_c, Y_c, Z_c) as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = T_{\text{RANSAC}} \cdot \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$

Visualization: Initial Pose Estimate (RANSAC only)

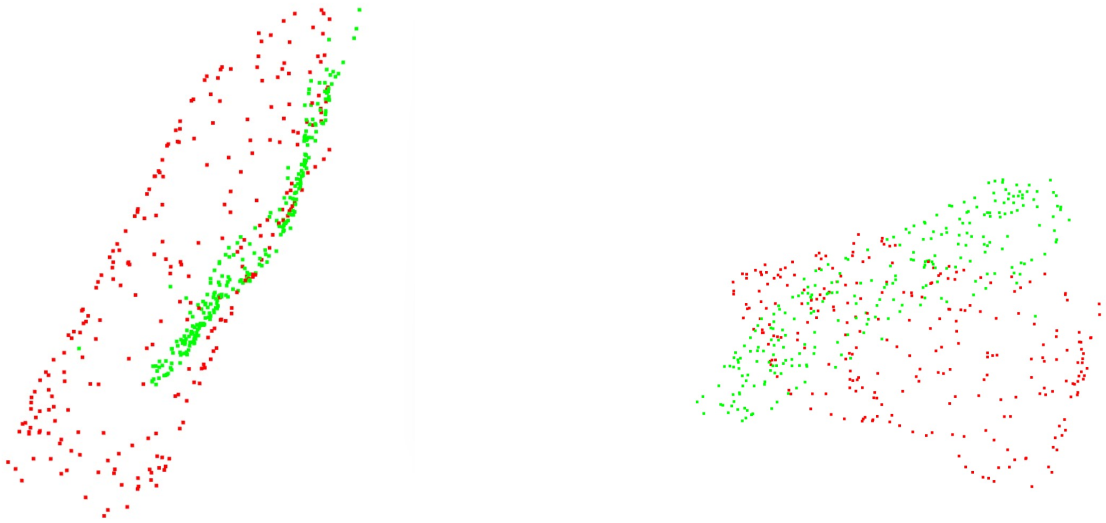


Figure 40: **RANSAC-Based Point Cloud Alignment.** The images depict the initial alignment between the CAD model (red) and the scene point cloud (green) using only the RANSAC algorithm. RANSAC identifies the best-fitting transformation by iteratively sampling correspondences and rejecting outliers. While the method provides a coarse estimate of the object pose, visible misalignments highlight the limitations of relying solely on sparse correspondences without geometric refinement.

8.1.2 Iterative Closest Point

While RANSAC provides a coarse alignment using robust matching of sparse keypoints, the result often contains errors in both rotation and translation due to noise or limited correspondences. To address this, we use the **Iterative Closest Point (ICP)** algorithm to refine the transformation.

Iterative Closest Point is a local optimization technique that minimizes the distance between corresponding points of the source and target point clouds. It starts with an initial guess (e.g., from RANSAC) and iteratively performs two steps:

1. Find the closest point in the target cloud for each point in the transformed source cloud.
2. Estimate the rigid transformation (R, t) that minimizes the following objective:

$$\min_{R,t} \sum_{i=1}^{N} \|R \cdot p_i + t - q_i\|^2$$

where:

- p_i are source cloud points,
- q_i are their corresponding target points,
- $R \in SO(3)$ is the rotation matrix,
- $t \in \mathbb{R}^3$ is the translation vector.

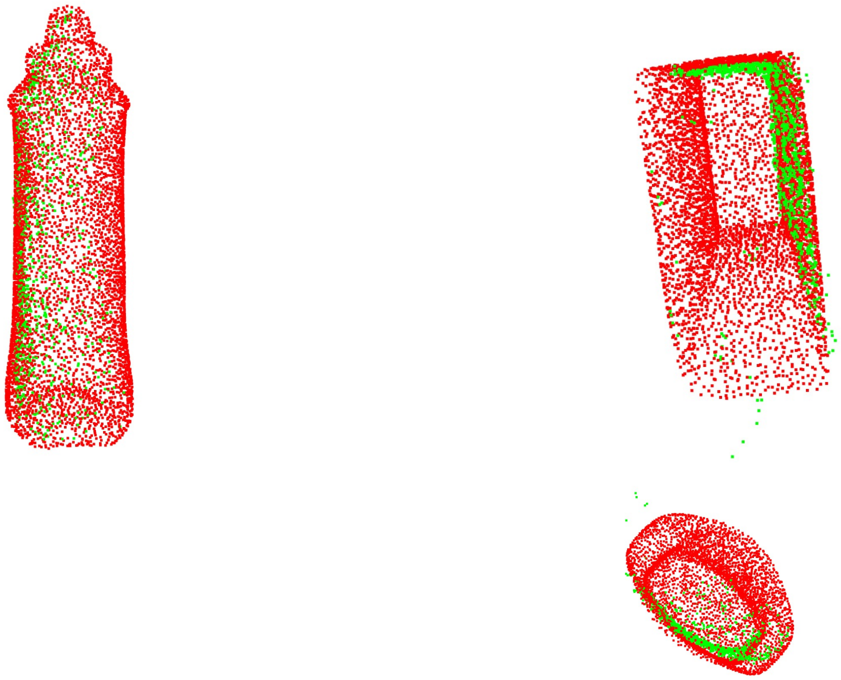


Figure 41: **ICP Refinement.** The red query cloud aligns more tightly to the green target after ICP, increasing geometric overlap—most visibly along edges and contours.

8.1.3 Singular Value Decomposition

After refining the coarse pose using ICP, we perform a final alignment using the **Singular Value Decomposition (SVD)** method to compute the optimal rigid transformation (rotation R and translation t) that best aligns the query point cloud $\mathcal{P}_Q$ to the target point cloud $\mathcal{P}_T$.

The objective is to minimize the mean squared error between corresponding points:

$$\min_{R,t} \sum_{i=1}^{N} \left\| R \cdot \mathbf{p}_i^Q + t - \mathbf{p}_i^T \right\|^2$$

where:

- $\mathbf{p}_i^Q$ and $\mathbf{p}_i^T$ are corresponding points from the query and target point clouds respectively,
- $R \in SO(3)$ is the rotation matrix,
- $t \in \mathbb{R}^3$ is the translation vector.

Algorithm

1. **Compute centroids**:

$$\bar{p}_Q = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i^Q, \quad \bar{p}_T = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i^T$$

2. **Center the point clouds**:

$$\mathbf{X} = [\mathbf{p}_1^Q - \bar{p}_Q, \ldots, \mathbf{p}_N^Q - \bar{p}_Q]$$

$$\mathbf{Y} = [\mathbf{p}_1^T - \bar{p}_T, \ldots, \mathbf{p}_N^T - \bar{p}_T]$$

3. **Compute the covariance matrix**:

$$H = \mathbf{X} \cdot \mathbf{Y}^\top$$

4. **Perform SVD**:

$$H = U \Sigma V^\top$$

5. **Compute optimal rotation and translation**:

$$R = V U^\top, \quad t = \bar{p}_T - R \cdot \bar{p}_Q$$

   If $\det(R) < 0$, the rotation is improper. In that case, we flip the sign of the last column of V:

$$V[:,3] \leftarrow -V[:,3], \quad R = V U^\top$$

8.1.4 Symmetry Aware Refinement

We use the Iterative Closest Point (ICP) algorithm to refine the coarse pose $\mathbf{T}_c$ at the point level and obtain a more accurate transformation $\mathbf{T}_f$. Although ICP is sensitive to local minima, it is effective when the initial pose is already close to the ground truth.

However, due to geometric symmetries, visually similar shapes can lead to ambiguous registrations even when using fused visual features. To address this, we introduce a symmetry-aware refinement stage that leverages rendering and patch-wise visual feature matching. This step corrects poses that are geometrically accurate but misaligned in texture — i.e., when $\mathcal{P}^Q$ and $\mathcal{P}^T$ match in shape but not appearance.

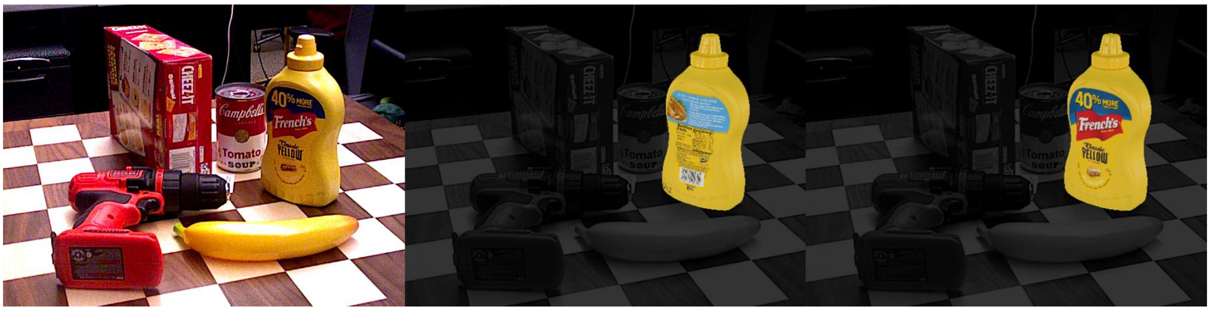


Figure 42: Symmetry-aware pose refinement. Left: input RGB image with initial pose. Middle: rendered view of the object after RANSAC + ICP. Right: refined pose with correct texture alignment—the SAR stage resolves rotational-symmetry ambiguities. *Click here to view the video demo of 6D pose estimation after SAR.*

Symmetry estimation. We aim to detect intrinsic geometric symmetries of the query object's mesh $\mathcal{M}^Q$, regardless of texture or color. We first sample a uniformly spaced set of rotations over $SO(3)$ by discretizing Euler angles in 45° increments along each of the x, y, and z axes. This results in a total of $S = 512$ candidate rotations $\{\mathbf{R}_s\}_{s=1}^{S}$.

The mesh is uniformly sampled into a point cloud $\mathcal{P}^Q \subset \mathbb{R}^3$ of N points. For each candidate rotation $\mathbf{R}_s$, we apply the transformation to the point cloud and compute a symmetric Chamfer distance between the rotated and original sets:

$$\mathcal{D}_{\text{Chamfer}}(\mathcal{P}^Q, \mathbf{R}_s\mathcal{P}^Q) = \frac{1}{N}\sum_{x\in\mathcal{P}^Q}\min_{y\in\mathbf{R}_s\mathcal{P}^Q}\|x-y\|^2 + \frac{1}{N}\sum_{y\in\mathbf{R}_s\mathcal{P}^Q}\min_{x\in\mathcal{P}^Q}\|y-x\|^2.$$

A rotation $\mathbf{R}_s$ is considered a valid symmetry if the distance falls below a threshold τ, i.e.,

$$\mathcal{D}_{\text{Chamfer}}(\mathcal{P}^Q, \mathbf{R}_s\mathcal{P}^Q) < \tau.$$

To ensure uniqueness of the discovered symmetries, we further discard matrices that are near-duplicates by checking their Frobenius norm difference:

$$\|\mathbf{R}_i - \mathbf{R}_j\|_F < \epsilon.$$

This approach enables automatic discovery of all self-aligning rotational symmetries of the object purely from geometry, independent of appearance or material properties.

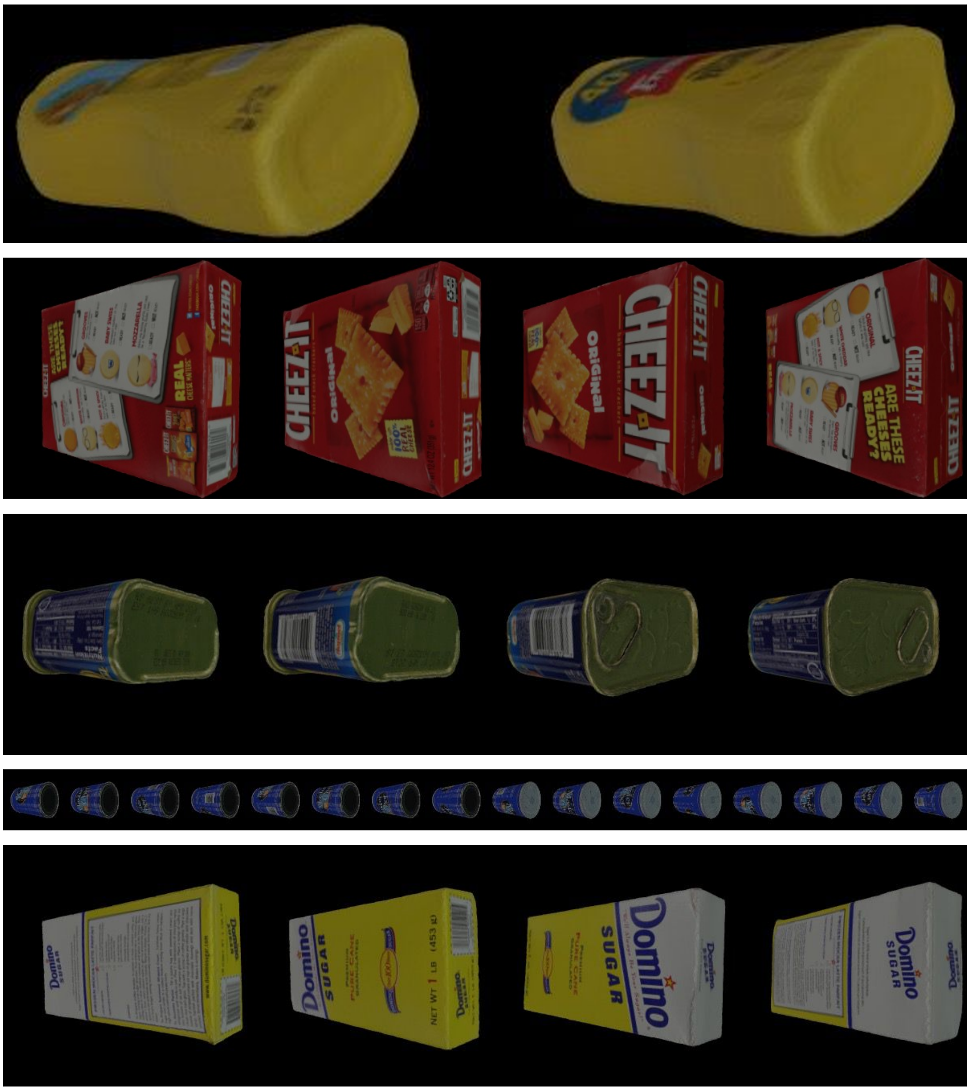


Figure 43: Visualization of multiple detected rotational symmetries for the same object. Each row corresponds to a different symmetry hypothesis

Symmetry-aware pose refinement. Given the estimated symmetry rotations $\{\mathbf{R}_s\}_{s=1}^{S}$ of the CAD model, we refine the predicted object pose by evaluating the perceptual alignment between rendered and observed image features.

For each candidate symmetry $\mathbf{R}_s$, we apply the composite transformation $\mathbf{T}_f \cdot \mathbf{R}_s$ to the canonical CAD model, where $\mathbf{T}_f \in SE(3)$ is the initially estimated object-to-camera pose. The resulting transformed mesh is rendered to produce an RGB image $\mathbf{I}_s$.

To compare the rendered and real images at the feature level, we crop both using a common 2D bounding box, yielding the views $\mathbf{I}_s^T$ and $\mathbf{I}^T$ for the symmetry candidate and target respectively. These cropped images are passed through a pretrained Vision Transformer encoder Φ to extract patch-wise descriptors:

$$\mathbf{V}_s = \Phi(\mathbf{I}_s^T), \quad \mathbf{V}^T = \Phi(\mathbf{I}^T)$$

where $\mathbf{V}_s, \mathbf{V}^T \in \mathbb{R}^{N \times D}$ denote sets of N patch embeddings of D dimensions.

The similarity between the symmetry candidate and the target image is evaluated using the mean cosine similarity over all patches:

$$\mathrm{Sim}(s) = \frac{1}{N} \sum_{i=1}^{N} \frac{\langle \mathbf{v}_i^T, \mathbf{v}_{s,i} \rangle}{\|\mathbf{v}_i^T\| \cdot \|\mathbf{v}_{s,i}\|}$$

The symmetry index s^* corresponding to the highest perceptual similarity is selected:

$$s^* = \arg\max_s \ \mathrm{Sim}(s)$$

Unlike typical global feature comparisons that rely on the class token in transformers, our method uses unpooled patch-level descriptors, which better preserve spatial and orientation-specific details, crucial for symmetry disambiguation.

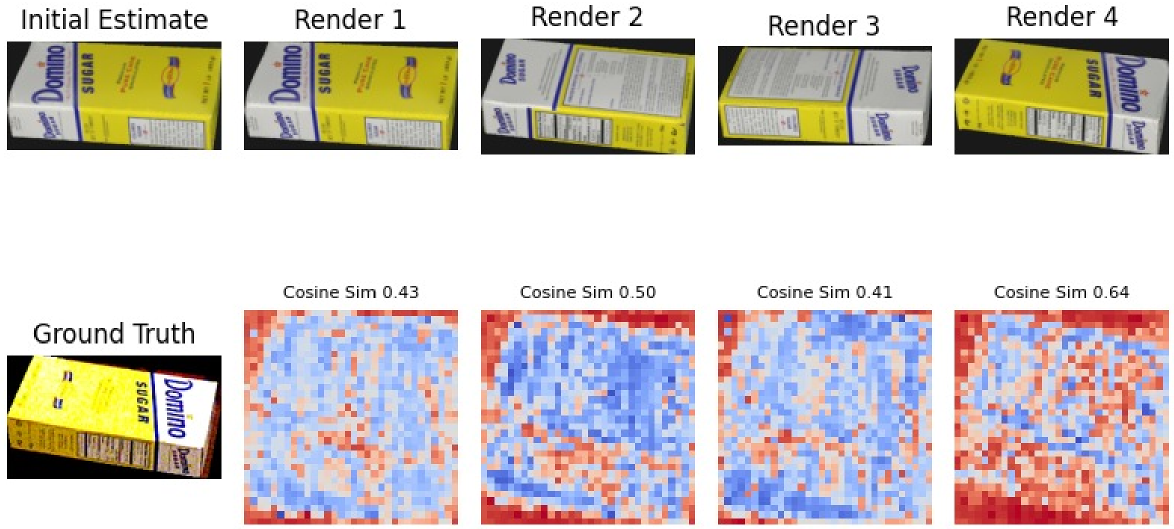


Figure 44: **Fig. 3: Example of symmetry-aware refinement.** We adjust the inaccurate initial pose (**top left**) by comparing the input crop (**bottom left**) with rendered images of the object's symmetric candidates (**first row**). **Sym. 0 (Id)** represents the identity transformation (no symmetry applied). By computing patch-level cosine similarities for each candidate (**second row**, red-white-blue), and averaging them to obtain a global score (**third row**), we select the best candidate as the one with the highest score.

8.1.5 Why We Choose RANSAC + ICP + Symmetry-Aware Refinement (SAR) Over Learned Models

Below, we summarize the justification for using the classical pipeline of **RANSAC + ICP + SAR** for pose refinement in FreeZe, contrasting it in detail with learned approaches such as GeoTransformer, CoFiNet, and OverlapPredator.

1. **Training-Free Philosophy:**
   The FreeZe pipeline adheres to a training-free, *zero-shot* design. All components, including RANSAC, ICP, and SAR, are classical algorithms and require no supervision or dedicated training on labeled data.
   In contrast, *learned models* like GeoTransformer, CoFiNet, and OverlapPredator depend on large labeled datasets and task-specific training, which contradicts the zero-shot refinement goal.

2. **Domain Generalization:**
   Classical methods are robust to large domain shifts, such as between synthetic CAD renderings and real RGB images, as found in FreeZe settings.
   Learned models often exhibit reduced generalizability and require careful domain adaptation to avoid performance drops in new environments.

3. **Modularity and Interpretability:**
   Algorithms like RANSAC, ICP, and SAR are *modular* and highly *interpretable*, simplifying debugging and deployment in real-time, containerized pipelines.
   Learned models tend to be *monolithic* and opaque, making them less adaptable and harder to interpret or trust in high-value or safety-critical deployments.

4. **Symmetry Handling:**
   SAR is specifically designed to disambiguate pose for symmetric objects by leveraging *rendering and feature matching* to resolve ambiguities left by purely geometric methods—a critical need for many industrial and household objects.
   Existing learned models lack integrated symmetry handling, often failing in cases with rotational or reflective symmetry.

5. **Efficiency and Deployment:**
   RANSAC + ICP + SAR are *computationally light* and supporting multi-threaded CPU execution. This makes them scalable for large deployments, even on resource-constrained hardware.
   Learned models are *heavier*, usually require GPU inference, and often bring latency and memory inefficiencies.

6. **No Dependence on External Matchers:**
   The FreeZe pipeline leverages dense features from *frozen foundation models* (e.g., DINOv2, GeDi) and requires no further external matching stages.
   Learned refiners like CoFiNet require an initial correspondence estimation step, thus increasing complexity and potential failure points.

9 Evaluation Metrics

To quantitatively evaluate the accuracy of our 6D object pose estimation system, we adopt standard metrics commonly used in pose estimation benchmarks: the **Average Distance of Model Points (ADD)** and its ambiguity-invariant variant **ADD-S**, along with translation and rotation errors computed directly from the transformation matrices.

ADD Metric. Given a 3D object model $\mathcal{S}$ consisting of N points, and the ground truth rotation R and translation T matrices, the ADD metric computes the average Euclidean distance between each 3D model point transformed using the estimated pose $(\hat{R}, \hat{T})$ and the ground truth pose (R, T). It is defined as:

$$\text{ADD} = \frac{1}{N} \sum_{x \in \mathcal{S}} \|(Rx + T) - (\hat{R}x + \hat{T})\|_2 \tag{7}$$

This metric is sensitive to object symmetries, meaning that if an object has indistinguishable symmetric views (e.g., cylindrical objects), even a visually correct pose may yield a high ADD.

ADD-S Metric. To address symmetry-related ambiguities, the ADD-S metric is used. Instead of directly comparing corresponding points, it computes the closest point distance from each transformed model point under the estimated pose to any transformed point under the ground truth pose:

$$\text{ADD-S} = \frac{1}{N} \sum_{x_1 \in \mathcal{S}} \min_{x_2 \in \mathcal{S}} \|(Rx_1 + T) - (\hat{R}x_2 + \hat{T})\|_2 \tag{8}$$

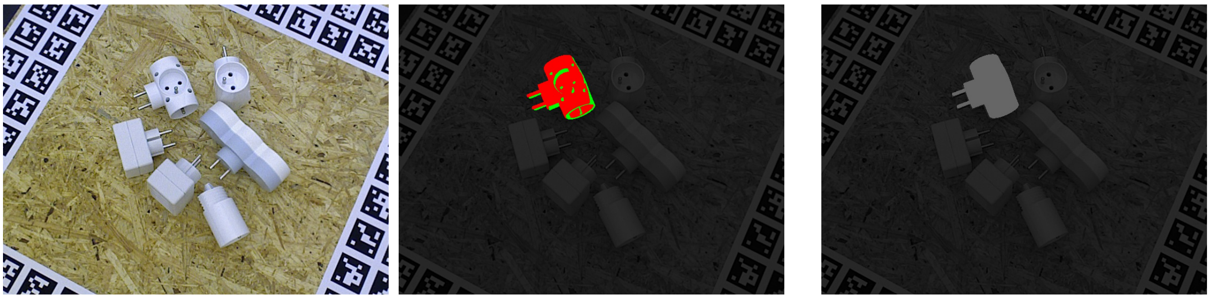
This makes ADD-S robust to both symmetric and non-symmetric object evaluations and is considered the primary metric for challenging real-world benchmarks.

Reprojection Error. In cases where RGB-only features are used for pose estimation, we may also report reprojection error. This is calculated by projecting the 3D model points onto the image plane using both the predicted and ground truth camera intrinsics and poses, and computing the average pixel distance between corresponding projections.

10 Qualitative Results

We present zero-shot 6D pose estimation results on the **T-LESS**, **YCB-Video**, and **LINEMOD** benchmarks.

10.1 T-LESS



(a) RGB image (b) Point Cloud Overlay (c) Pose Estimation

Figure 45: Qualitative results on the **T-LESS** dataset. Predicted pose is visualised in red, ground-truth in green.

10.2 LINEMOD

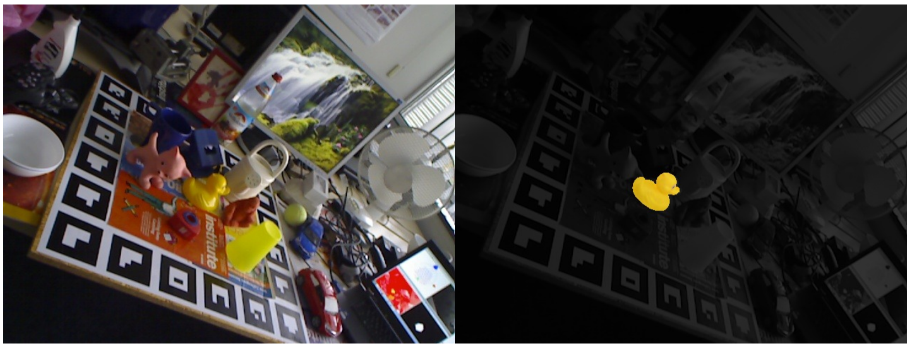
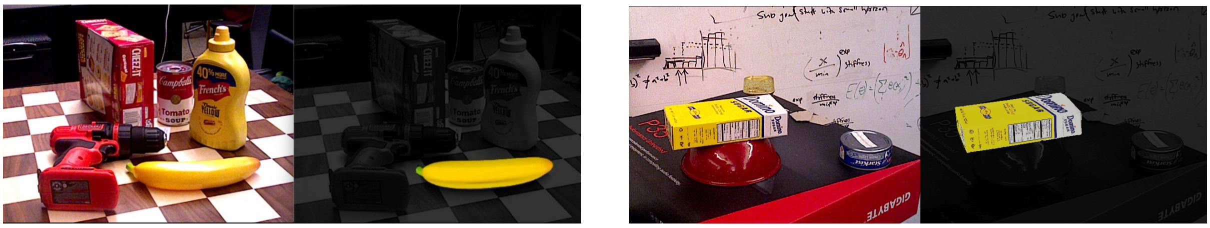


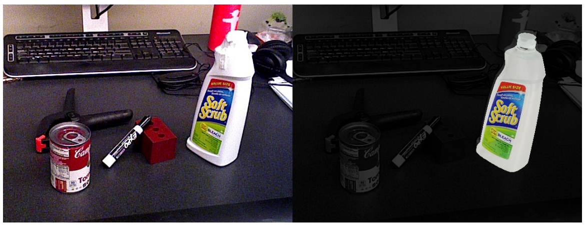
Figure 46: Qualitative result on **LINEMOD**. The yellow overlay indicates the recovered rubber-duck pose.

10.3 YCB-Video



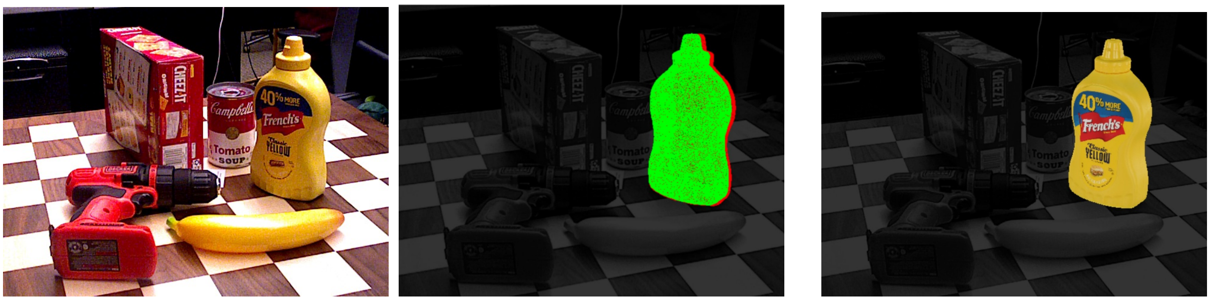
(a) Banana

(b) Domino Sugar Box



(c) Soft Scrub Bottle

Figure 47: Qualitative results on **YCB-Video**.



(a) RGB image

(b) Point Cloud Overlay

(c) Pose Estimation

Figure 48: Predicted pose is visualised in red, ground-truth in green.

11 References

1. Caraffa, A., Boscaini, D., Hamza, A., Poiesi, F. *FreeZe: Training-Free Zero-Shot 6D Pose Estimation with Geometric and Vision Foundation Models.* ECCV 2024. https://arxiv.org/pdf/2312.00947v3.pdf
2. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V. *Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer.* arXiv 1907.01341 (2019). https://arxiv.org/pdf/1907.01341v2.pdf
3. Kirillov, A., Mintun, E., Ravi, N., *et al.* *Segment Anything.* ICCV 2023. https://arxiv.org/pdf/2304.02643.pdf
4. Radosavovic, I., Caba, Y., *et al.* *DINOv2: Self-Supervised Vision Backbone.* Meta AI, 2023. https://arxiv.org/pdf/2304.07193
5. Poiesi, F., Boscaini, D. *Learning General and Distinctive 3-D Local Deep Descriptors for Point Cloud Registration (GeDi).* arXiv 2105.10382 (2022). https://arxiv.org/pdf/2105.10382v3.pdf

6. Huang, S., Gojcic, Z., Usvyatsov, M., Wieser, A., Schindler, K. *PREDATOR: Registration of 3-D Point Clouds with Low Overlap.* CVPR 2021. https://arxiv.org/pdf/2011.13005.pdf
7. Nguyen, V.N., Groueix, T., Ponimatkin, G., Lepetit, V., Hodan, T. *CNOS: A Strong Baseline for CAD-Based Novel Object Segmentation.* ICCV W 2023. https://arxiv.org/pdf/2307.11067
8. Caraffa, A., Boscaini, D., Poiesi, F. *Accurate and Efficient Zero-Shot 6D Pose Estimation with Frozen Foundation Models.* arXiv 2506.09784 (2025). https://arxiv.org/pdf/2506.09784
9. Zhang, C., Liwicki, S., Smith, W., Cipolla, R. *Orientation-Aware Semantic Segmentation on Icosahedron Spheres.* ICCV 2019. https://openaccess.thecvf.com/content_ICCV_2019/papers/Zhang_Orientation-Aware_Semantic_Segmentation_on_Icosahedron_Spheres_ICCV_2019_paper.pdf
10. Örnek, E.P., Labbé, Y., Tekin, B., *et al. FoundPose: Unseen Object Pose Estimation with Foundation Features.* arXiv 2311.18809 (2024). https://arxiv.org/pdf/2311.18809
11. Ausserlechner, P., Haberger, D., Thalhammer, S., *et al. ZS6D: Zero-Shot 6D Object Pose Estimation Using Vision Transformers.* arXiv 2309.11986 (2023). https://arxiv.org/pdf/2309.11986
12. Qi, C.R., Su, H., Mo, K., Guibas, L.J. *PointNet: Deep Learning on Point Sets for 3-D Classification and Segmentation.* CVPR 2017. https://arxiv.org/pdf/1612.00593
13. Zhou, R., *et al. GigaPose: Fast and Robust Novel Object Pose Estimation via One Correspondence.* CVPR 2024. https://openaccess.thecvf.com/content/CVPR2024/html/Zhou_GigaPose_Fast_and_Robust_Novel_Object_Pose_Estimation_via_One_Correspondence_CVPR_2024_paper.html
14. Yu, W., *et al. PoinTr: Diverse Point-Cloud Completion with Geometry-Aware Transformers.* ICCV 2021. https://arxiv.org/abs/2108.01486
15. Labbe, Y., Aubry, M., *et al. MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare.* arXiv 2212.06870 (2022). https://arxiv.org/abs/2212.06870
16. Gojcic, Z., Thalhammer, S., *et al. FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects.* arXiv 2312.08344 (2023). https://arxiv.org/abs/2312.08344
17. Rusu, R.B., Blodow, N., Beetz, M. *Fast Point Feature Histograms (FPFH) for 3D Registration.* ICRA 2009. http://rusu.info/files/icra2009.pdf
18. Choy, C., Park, J., Koltun, V. *Fully Convolutional Geometric Features for Keypoint Matching in 3D Point Clouds.* ICCV 2019. https://arxiv.org/pdf/1904.12681.pdf
19. Li, Y., *et al. Unified Category-Level 6D Pose Estimation With Foundation Features.* arXiv 2308.03768 (2023). https://arxiv.org/abs/2308.03768
20. Zhang, Y., *et al. FoundAlign: Foundation Models Meet Object Alignment.* arXiv 2402.02217 (2024). https://arxiv.org/abs/2402.02217
21. Liu, S., *et al. Latent-Diffusion for 6D Pose: Geometry-Aware Diffusion Models.* arXiv 2202.06688 (2022). https://arxiv.org/abs/2202.06688
22. Lee, H., *et al. S2P3: Self-Supervised Pose Prior for Novel-Object 6D Localization.* arXiv 2405.17859 (2024). https://arxiv.org/abs/2405.17859

23. Wang, Y., *et al.* *CAD Fusion: Learning 6D Pose With CAD Supervision Alone.* arXiv 2105.14021 (2021). https://arxiv.org/abs/2105.14021

24. Gümüş, B., *et al.* *Open-Set 6D Object Pose via Diffusion Priors.* arXiv 2405.14458 (2024). https://arxiv.org/abs/2405.14458

25. Ranjan, V., *et al.* *DiffPose: Diffusion Models for Category-Level Pose.* arXiv 2408.09332 (2024). https://arxiv.org/html/2408.09332v1

26. Patel, K., *et al.* *Image-To-CAD Alignment With Cross-Modal Transformers.* arXiv 2307.14030 (2023). https://arxiv.org/abs/2307.14030

27. Hodan, T., *et al.* *6-D Localization of Unseen Objects – Core BOP 2023 Datasets.* Online leaderboard, accessed 01 Mar 2024. https://bop.felk.cvut.cz/leaderboards/pose-estimation-unseen-bop23/core-datasets/

28. Goodwin, W., Vaze, S., Havoutis, I., Posner, I. *Zero-Shot Category-Level Object Pose Estimation.* ECCV 2022.

29. Qi, C.R., Yi, L., Su, H., Guibas, L.J. *PointNet++: Deep Hierarchical Feature Learning on Point Sets.* NeurIPS 2017. https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d30a8f07f18d2b-Paper.pdf

30. Rad, M., Oberweger, M., Lepetit, V. *Feature Mapping for Fast and Accurate 3-D Pose Inference From Synthetic Images.* CVPR 2018.

31. Zhuang, C., Li, S., Ding, H. *Instance-Segmentation-Based 6D Pose Estimation of Industrial Objects for Robotic Bin-Picking.* Robotics and Computer-Integrated Manufacturing (2023).

32. Wang, C., *et al.* *DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion.* CVPR 2019.

33. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D. *PoseCNN: A CNN for 6D Object Pose Estimation in Cluttered Scenes.* RSS 2018.

34. Sun, J., *et al.* *OnePose: One-Shot Object Pose Estimation Without CAD Models.* CVPR 2022.

35. Hodaň, T., Haluza, P., Obdržálek, Š., *et al.* *T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects.* WACV 2017 / ECCV 2018. https://cmp.felk.cvut.cz/t-less/

36. Chen, X., *et al.* *ZeroPose: CAD-Prompted Zero-Shot Object 6D Pose Estimation in Cluttered Scenes.* IEEE TCSVT 2024. https://arxiv.org/abs/2402.02217

37. Moon, G., *et al.* *GenFlow: Generalizable Recurrent Flow for 6D Pose Refinement of Novel Objects.* CVPR 2024. https://arxiv.org/abs/2405.17859

38. Zhao, Y., Li, C., *et al.* *Learning Symmetry-Aware Geometry Correspondences for 6D Object Pose Estimation.* ICCV 2023. https://arxiv.org/abs/2308.03768

39. Qin, Y., Li, R., *et al.* *Geometric Transformer for Fast and Robust Point Cloud Registration.* CVPR 2022. https://arxiv.org/abs/2202.06688

40. Besl, P.J., McKay, N.D. *A Method for Registration of 3-D Shapes.* IEEE TPAMI 1992.

41. Arun, K.S., Huang, T.S., Blostein, S.D. *Least-Squares Fitting of Two 3-D Point Sets.* IEEE TPAMI 1987.

42. Sun, J., *et al.* *SAM-6D: Segment Anything Model Meets Zero-Shot 6D Object Pose Estimation.* CVPR 2024. https://arxiv.org/abs/2405.14458